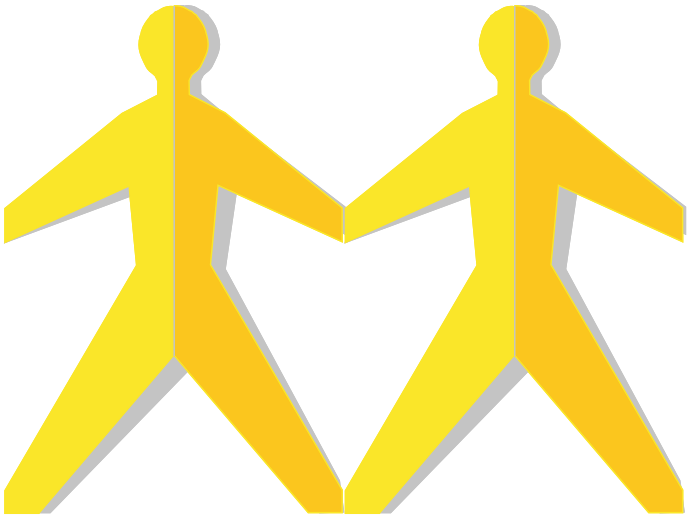
gedas



# rvsEVO

Version 3.0

User Manual

rvsEVO
User Manual


Pascalstraße 11
10587 Berlin

## Contents

# 1 Introduction

In this chapter you will find a short description of rvs® and rvsEVO as well as an explanation of typographic conventions used in the present manual.

## 1.1 Short description of the system

### What is rvs®

rvs® = Rechner-Verbund-System

The abbreviation rvs® stands for the German word Rechner-Verbund-System. The rvs® computer communication system is a well established base service for electronic data interchange, EDI.

rvs® serves to ensure transmission of electronic data between heterogeneous computer platforms using different network protocols.

To do so, rvs® implements a universal network model, which you can configure in each network node.

rvs® provides an efficient and reliable transport service for both standardized EDI message types and files of any format or contents. You can receive only such files that are explicitly destined for rvs®. This means that rvs® does not allow any unauthorized access to remote or to own data files.

The system was originally developed by Volkswagen AG and has been used in the German and European automobile industries for a number of years but also by banks, insurances and industry worldwide.

rvs® uses the OFTP protocol. An extension to the OFTP standard was developed for Volkswagen AG: It has been enhanced by a line driver for SNA LU 6.2.

### What rvs® is not

rvs® is not an online system. It neither supports direct terminal-like access to other sites, nor does it provide a communication pipe from application to application on a data record level. You cannot directly execute transfers in your own application. You rather can place send orders from within you application to rvs® which will be handled asynchronously.

rvs® is not a job scheduling system.

rvs® does not care about the contents of the files it is transporting. It only acts as a transparent transport medium and performs no semantic interpretation of the data it carries.

rvs® is not an EDI converter. You can, however, purchase additional components for converting between specific message formats (e.g. VDA, ODETTE, EDIFACT, XML) using rvs® as transport service from gedas deutschland GmbH.

rvs® is not a network control or monitoring tool.

### What is rvsEVO

rvsEVO

rvsEVO is a communication software with a graphical user interface based, like rvs®, on the OFTP protocol.

The present product adds a number of new functions to the existing rvs® Tiny product (see below). As this is an extension of rvs® Tiny version 2.0, rvsEVO starts with version number 3.0.

available features

The following features are available in rvsEVO 3.0:

- send files to the neighbour or to the routed stations.
- receive files from the neighbour or from routed stations .
- activate the the neighbour stations
- view information about receive jobs, send jobs, failed jobs and ended jobs.
- display monitor messages
- delete or release EERPs (End-to-End-Response) if necessary
- archive information about the processes send or receive jobs in the revision log.
- define job filters and actions when sending or receiving files
- code conversion with various code conversion tables
- format conversion
- log files for tracing the Monitor activities and for troubleshooting
- Compression and Encryption
- Backup and Recovery
- Transmission of large files up to 10 GB
- Support of Central Journal functions (for more information see the Central Journal User Manual)
- Support of SNMP Monitoring (for more information see the rvs® SNMP Agent User Manual)

**Note**: rvs® Tiny can have only one neighbour station: rvs® center. The following features are not supported in rvs® Tiny: Compression and Encryption; Backup and Recovery; Central Journal and SNMP Monitoring.

rvsEVO uses a batch interface and the file system to communicate with the application. If capable to do so, the linked application can indicate successful processing and have successful dispatch indicated.

rvsEVO is implemented in Java.

TCP/IP protocol

At present, rvsEVO only supports the OFTP TCP/IP protocol.

For more information on supported platforms please refer to the `$RVS_HOME\doku\liesmich.txt` release notes.

**Note**: Please read the chapter 1.2 for the explanation of `$RVS_HOME`.

## 1.2 Representation means

This chapter describes the typographic conventions used in this manual and explains the meaning of specially highlighted expressions.

### Typographic conventions

- Instructions begin with a bullet.

– Other lists begin with the en dash.

Character styles

| | |
|---|---|
| `Courier` | Commands, menu commands, file names, path names, programs, examples, scripts, options, qualifiers, data sets, fields, modes, window names, dialog boxes and statuses |
| **BOLD** and **IN CAPITAL LETTERS** | Parameters, environment variables, variables |
| “Inverted commas” | Links to other manuals, sections and chapters, literature |
| **Bold** | Important terms, names of operating systems, proper names, buttons, function keys. |

### Directories

**$RVS_HOME**

As user directories are found on different locations for the different operating systems we use the variable **$RVS_HOME** in this manual. Default values are:

– `C:\Programs\rvsEVO` for **Windows XP** and **Windows 2000**

Substitute the variable with your correct path.

## 1.3 Target group

This manual is meant for regular users of rvsEVO as well as administrators. It provides an overview of the basic rvsEVO functions.

Skills

The following skills are required to be able to use rvsEVO:

– good knowledge of the current operating system

– knowledge of the communications techniques in use TCP/IP.

Before starting to work with rvsEVO it is advisable to have read this book.

# 2 Installation

The present chapter describes the system requirements as well as the rvsEVO installation procedure.

## 2.1 System requirements

To successfully operate rvsEVO you need the following software:

Software

- Operating system: Windows XP or Windows 2000, UNIX (AIX, SunOS, HP-UX, Linux) or OpenVMS.
- Java runtime environment (JRE 1.4._XX or Java Software Development Kit 1.4_XX).

  Please make sure a Java runtime environment is present on your system prior to installing rvsEVO. The software is freely available for download from http://java.sun.com.

Initially, you need at least 8 MB free space on your hard disk. Depending on the amount of usage, the retention period for old entries, and the time between database cleanups, the space requirement may be considerably larger.

At present, rvsEVO only supports the OFTP TCP/IP protocol.

As a rule rvsEVO ships on CD-ROM or magtape. Your system must be able to read these media. Please contact your distributor if you have different requirements.

## 2.2 Obtaining a license

You need a license key to work with rvsEVO.

rvs® after-sales service

Please contact the rvs® after-sales service (phone: +49 30 39971 777; fax: +49 30 39971 994; email: rvs-service@gedas.de) to receive a license key.

To purchase a license key:

- Type `hostname` in the command prompt window (`Run -> cmd`).
- Send the command output (your computer name, e.g. `BWcd00034`) to the rvs® after-sales service. Please make sure to heed upper- and lowercases as otherwise a valid license key for your computer cannot be generated.
- You will be sent your license key file by email.

Save the license key file in the `$RVS_HOME\conf\` directory as `license.properties`.

**Note**: Please read the chapter 1.2 for the explanation of `$RVS_HOME`.

## 2.3 Fresh install of rvsEVO

Installation steps

This chapter describes the installation of rvsEVO. Please read the chapter 2.1 „System requirements“ before installing.

First we describe how to perform installation on Windows systems. Then we briefly cover installation on UNIX systems because installation is identical on both operating systems.

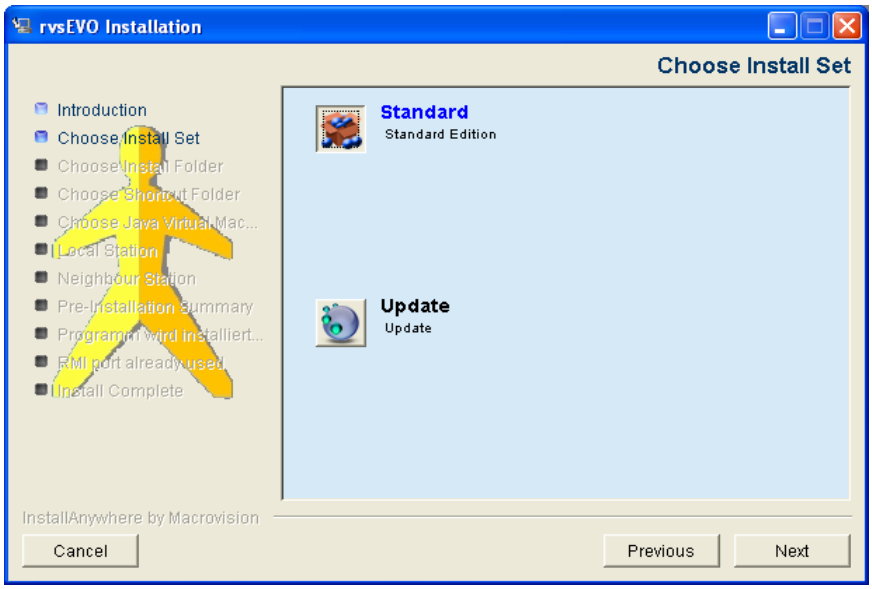
OpenVMS

A separate document covers installation on OpenVMS systems; it is available upon request from your sales partner (phone: +49 30 39971 537; fax: +49 30 39971 994; email: rvs-sales@gedas.de) or the rvs® customer service (phone: +49 30 39971 777; fax: +49 30 39971 994; email: rvs-service@gedas.de).

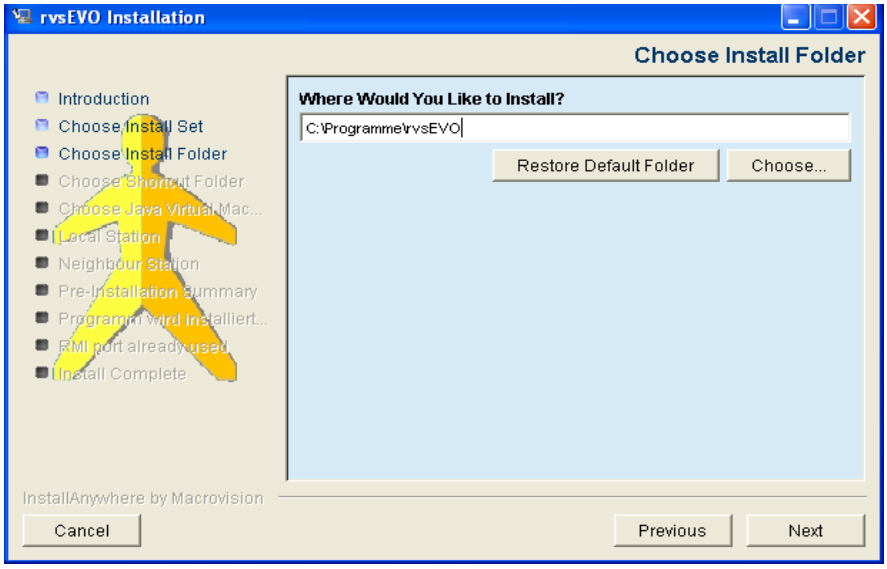
### Installation on Windows Systems

- Start Windows and log in as a Windows user with administrator rights.
- Start the `rvsEVO_X.X_setup.exe` installer (where `X.X` indicates the rvsEVO version number) by double-clicking or using the Windows command: `Start -> Run`.
- Choose the installation language (German, English) in the first dialog. Click **<OK>** to go to the next installation step.

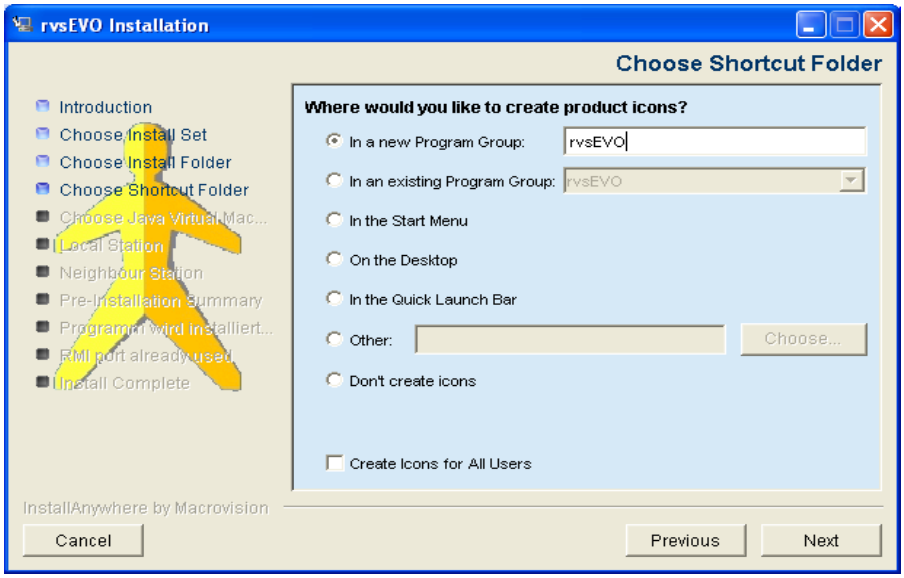

- In the next dialog you can choose between a “standard” installation and an update.



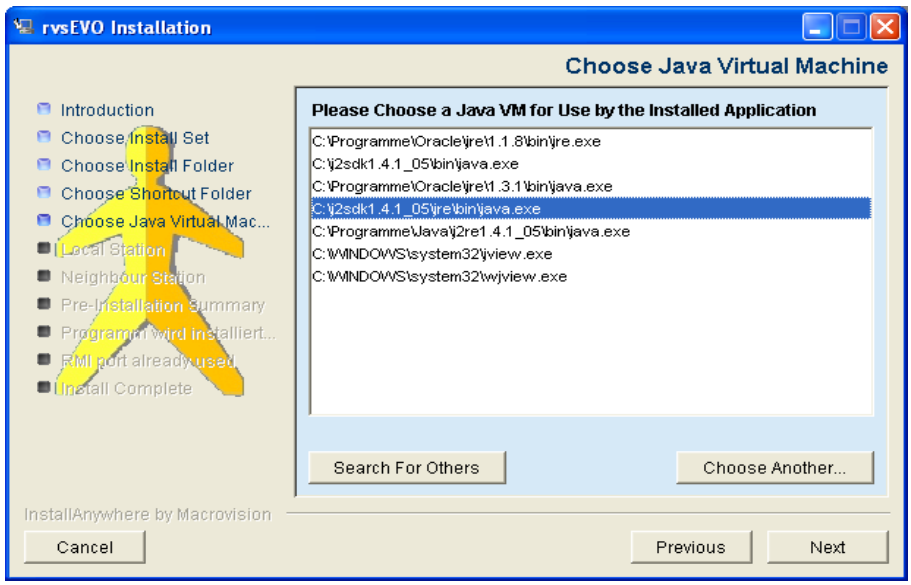

- In the next dialog you can select the rvsEVO destination directory.



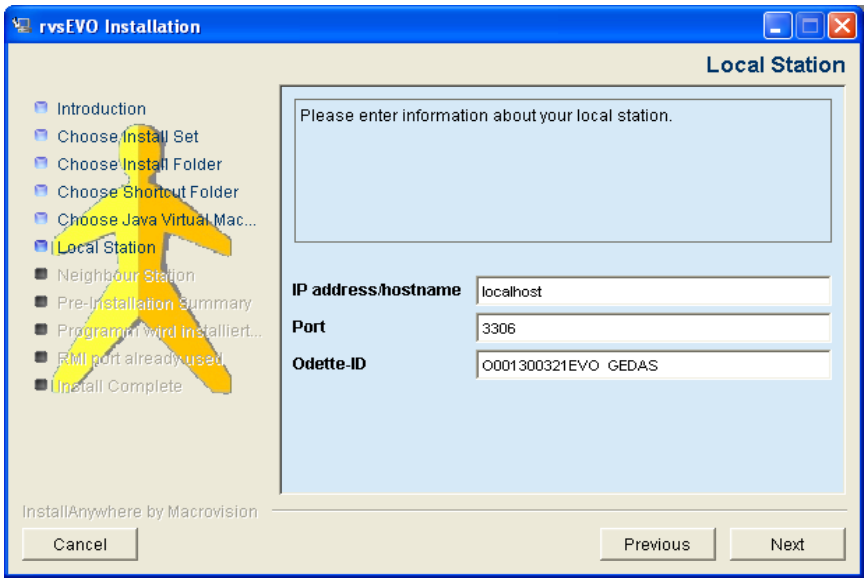

- Define the program group to install rvsEVO icons to in the following dialog.



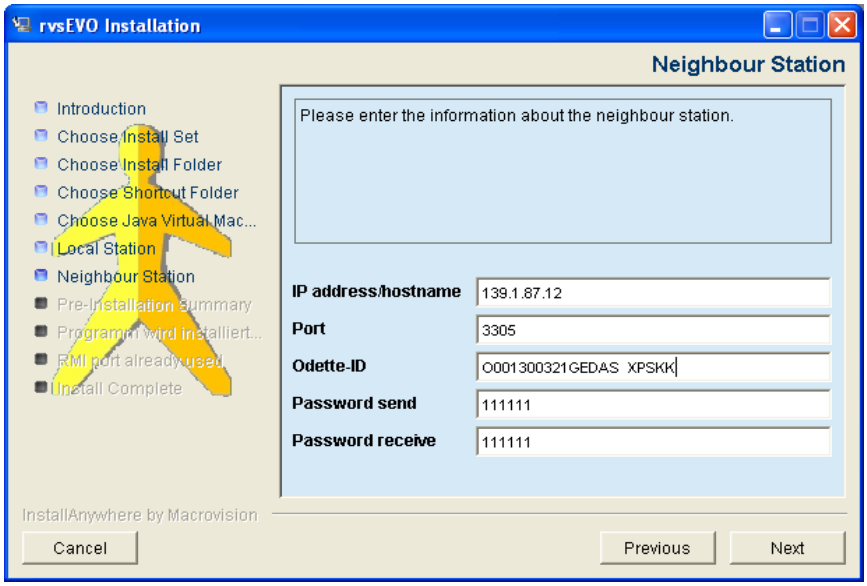

- Define the Java runtime environment for rvsEVO operation in the next dialog. The installer searches for installed components and proposes the versions found in a dialog. rvsEVO has been released for use from version 1.4._X onwards.




- The following dialogs deal with the settings in the station list. You must first configure the entry for the local station. The three parameters in the dialog are mandatory and must be specified. Please read the chapter 3.2 for more information about setting station parameter (e.g. how to obtain the ODETTE ID).




- Then follows the dialog for the direct neighbor station parameters. All of these parameters are mandatory as well and must be specified to ensure proper operation of rvsEVO.




- In the next dialog you are given a brief overview of selections you have made (installation directory, link directory). The required and the currently available disk space is also indicated. Press the **Install** button to start installation and to copy the installation files into the directories you specified.
- The last dialogs informs you of the successful installation of rvsEVO.

### Installation on UNIX Systems

As mentioned earlier in this chapter, installation on UNIX systems runs analog to an installation on Windows systems. The installation file is named `rvsEVO_X.X.X_setup.bin` and can be started as a window-based installation under the X-Server or in the console mode with the `-console` option.

**Note**: Make sure to call the installation file as a shell script when you perform installation in the console mode.

**Example** (command line):
`sh ./rvsEVO_300_00_SE_setup.bin -console`

The installation prompts are identical in both modes (see section **Installation on Windows systems**).

There are minor differences for particular UNIX platforms mentioned in the release notes for the respective version (`$RVS_HOME\doku\liesmich.txt` document).

## 2.4 How to start rvsEVO?

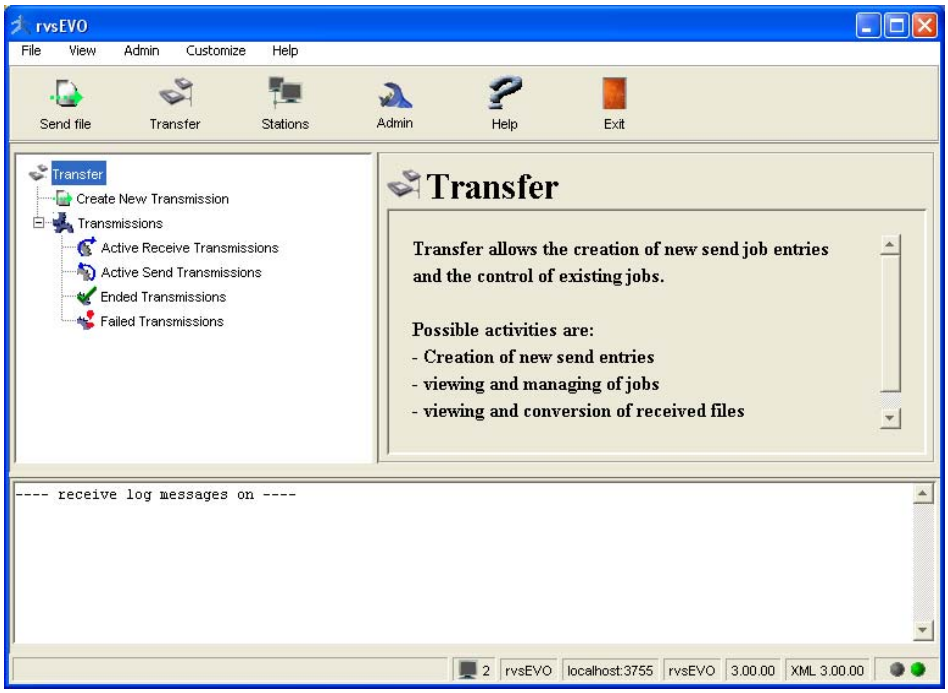
**Windows**: Start rvsEVO by choosing `Start -> Programs -> rvsEVO -> startGUI` (or the Start menu command you specified during installation) in the Start menu.

**Unix**: Start rvsEVO by starting the shell script `$RVS_HOME\bin\startGUI.sh.`

This menu item (Program) first starts the user interface, which then starts the rvsEVO server.

### Successful start:

Start A successful start is indicated as follows:




The `startGUI` program starts the user interface; to start the rvsEVO server, use the `startService` or `startServer` script on Windows systems and `startServer` on UNIX systems. These scripts are located in the `$RVS_HOME/bin/` directory (on Windows systems as batch files, on UNIX systems as shell scripts.

**Note**:

**Windows**: By default, rvsEVO starts as a service on Windows systems. You can configure this in the `$RVS_HOME/config/rvsConfig.xml` file using the `RvsStartScript` parameter (see also chapter 3.1).

As an alternative you can also start rvsEVO at the command prompt by defining the `$RVS_HOME/bin/startServer.bat` program as the `RvsStartScript` program.

**Unix**: By default, the `$RVS_HOME/bin/startServer.sh` shell script is used as the `RvsStartScript` on Unix systems.

## 2.5 How to stop rvsEVO?

**Windows**: Stopt rvsEVO by choosing `Start -> Programs -> rvsEVO -> stop Server` (or the Start menu command you specified during installation) in the Start menu.

As an alternative you can also stop rvsEVO at the command prompt by the `$RVS_HOME/bin/stopServer.bat` program.

**Unix**: Stop rvsEVO by starting the shell script `$RVS_HOME\bin\stopServer.sh`.

### 2.6 rvsEVO as Windows service

Per default is rvsEVO installed as a Windows service.

**Note**: The term Service means a program that can be started from the operating system and works in the background.

This is possible starting the batch script `rvsservice.bat` out of the directory `$RVS_HOME\bin`.

**Usage:**

```
rvsservice <parameters> [options]
```

The possible parameters are:

| | |
|---|---|
| **-c** | starts rvsEVO on console |
| **-i** | installs rvsEVO as service |
| **-r** | removes rvsEVO as service |
| **-s** | starts rvsEVO as service |
| **-h** | usage |

**Example**:

```
C:\Programme\rvsEVO\bin>rvsservice -i
-------------------------
-- RVSTINY SERVICE LAUNCHER --
-------------------------
rvs Server installed.
```

With the command `rvsservice -c` rvsEVO Server was started on the command line (console).

With the command `RVSservice -i` rvsEVO was installed as system (Windows) service.

Now you can find rvsEVO as a service in the list of system services (`Start - > Control Panel -> Administrative Tools -> Services`). If you want to start rvsEVO every time the system starts, you can set the startup type to `Automatic` by choosing Automatic from the combo box in the **Startup type** area.

**Example** (Windows XP german version):

## 2.7 rvsEVO update installation

An rvsEVO update installation is almost identical to a normal installation (see chapter 2.3 "Fresh install of rvsEVO").

Please read chapter 7 to learn how you can use the Central Administration to update other rvsEVO installations.

# 3 Configuration

The present chapter describes how to customize configuration files via the rvsEVO GUI or via rvsEVO command line.

**Note**: You do not need to make any changes to obtain a functioning system if you have correctly specified all parameters during installation. You must restart rvsEVO each time you have made changes to a configuration file.

## 3.1 Customizing the global configuration files

Files Two configuration files are of vital significance for the global rvsEVO settings.

– `$RVS_HOME\conf\rvsConfig.xml`
– `$RVS_HOME\conf\rvs.properties.`

### 3.1.1 rvs.properties

This is a Java properties file containing entries of the following format:

`<name>=<value>.`

This file refers to 3 important locations (contains three important parameters):

– The rvsEVO installation directory (parameter: `RootDir`)
– The global rvsEVO configuration file (parameter: `ConfigFile`)
– The global configuration file for system logging (parameter: `LogConfigFile`).

Sample file (comment lines starting with # will be ignored):

```
RootDir=C:\\Programs\\rvsEVO
# configuration xml file
ConfigFile=C:\\Programs\\rvsEVO\\conf\\rvsConfig.xml
# log config file
LogConfigFile=C:\\Programs\\rvsEVO\\conf\\rvsLogger.xml
```

### 3.1.2 rvsTinyConfig.xml

Elements The following table lists the most important elements of the `rvsConfig.xml` file.

| ELEMENT | DESCRIPTION |
|---|---|
| **AgentActive** | This parameter defines whether or not rvs® SNMP Agent is enabled or disabled: Default: `Y` (Yes). Possible values: `Y` (Yes) or `N` (No). |

| ELEMENT | DESCRIPTION |
|---|---|
| **Agent-HeartbeatInterval** | This parameter defines the interval (in seconds) at which rvsEVO sends a Heartbeat message to the Agent UDP address (AgentHostname + AgentPort). |
| **AgentHostname** | Agent computer name (or IP address). Default: `localhost`. |
| **AgentPort** | Agent IP port. Default: `3744`. |
| **AgentLogLevel** | This parameter defines whether or not rvsEVO sends log messages to the Agent. Possible values: 0, 1.<br>0: no log messages are sent.<br>1: all log messages are sent. |
| **BackupStartup** | You can use this parameter to specify an automatic back-up to be performed each time you start rvsEVO. Possible values: `Y` (Yes); `N` (No): The default is `Y`. |
| **Cleanupdays** | Specify days for archiving of completed jobs. The `archiveJobs` program will save any jobs older than the time specified in this parameter to the `RevisionLog.xml` file (see chapter 4.9, see also the `PersistenceArchive` parameter). |
| **Cleanupinterval** | Time interval in hours between two archiving cycles started with the `Cleanupdays` parameter. |
| **CentralJournalInstance** | The rvs® destination station that is to receive the Journal files. This station must be present in the rvsEVO station list. |
| **JournalFilenamePrefix** | The prefix for the Journal file name: default `TINY`. |
| **SendJournalInterval** | Time interval in seconds between sending of two Journal files to the rvs® destination station (defined by the `CentralJournalInstance` parameter). No Journal file will be sent if no value or 0 is specified here. |
| **ConnSetupFailWaitTime** | Time in milliseconds rvsEVO waits after a connection failed to be established before rvsEVO tries to establish the connection again. |
| **JobAfterSECreation** | Script that is to be started as soon as the send entry was generated. This parameter is optional and contains the name of a script that must be located in the `$RVS_HOME/bin` directory. See also chapter 4.5 "Sending a file" (`CreateSendJob`). This script receives the TransmissionID as the first parameter. |
| **LooptestNeighbourSID** | ID of the station via which the loop test (transmission of a file to the own local station) is performed. The file is to be transmitted to the own station, and rvsEVO sends this file via a neighbor station back to the local station. |

| ELEMENT | DESCRIPTION |
|---|---|
| **MaxSessions** | Maximum number of simultaneously running receiving processes for TCP/IP communication. Default: 2; maximum number is restricted by system resources. |
| **DB** | Directory for job administration with the `ENDED`, `FAILED`, `RCV` and `SND` subdirectories. The `RCV` and `SND` subdirectories are used to store temporary, not fully processed jobs. The `FAILED` directory holds the failed, the `ENDED` directory the completed jobs. These directories are also visible in the GUI (Transfer window, Transmissions). |
| **TEMP** | Directory for temporary use. |
| **INBOX** | Directory where completely received files are stored. |
| **OUTBOX** | Temporary directory for files to be sent. |
| **ARCDIR** | Archive directory where files such as the Revision log are stored. |
| **LOGDIR** | Directory for both log files: `monlog.log` and `tiny.log`. |
| **JobstartConfigFile** | Configuration file for job start. |
| **MaxMonLogCount** | Number of `$RVS_HOME/log/monlog.log` log files that can be generated. The file with the greatest number is the one generated last. |
| **MaxMonLogSize** | Maximum file size of `monlog.log` log file (in bytes) |
| **MaxRevisionLog-Count** | Number of `RevisionLog.xml` revision files that can be generated. Refer to chapter 4.9 to learn how to generate a revision file. See also `PersistenceArchive` parameter in this table. |
| **MaxRevision-LogSize** | Maximum file size of the `RevisionLog.xml` revision file (in bytes) |
| **OFTPTimeout** | Time-out in milliseconds at ODETTE level; default: 300 000, no maximum. |
| **RvsStartScript** | Path of the script staring rvsEVO; default: `$RVS_HOME/bin/startServer.bat` or as Windows service: `$RVS_HOME/bin/startService.bat` |
| **StationsConfigFile** | Station configuration file: Comprises the configuration parameters for the local station, the neighbor station, and routed stations. |
| **HostAllowFile** | Configuration file containing DNS names or IP addresses of hosts that may send rvsEVO commands to the rvsEVO server. |

| ELEMENT | DESCRIPTION |
|---|---|
| **HostDenyFile** | Configuration file containing DNS names or IP addresses of hosts that may not send rvsEVO commands to the rvsEVO server. |
| **SessionAliveTimeout** | Time in milliseconds to consider a connection active; default 600 000, no maximum. |
| **Timestamp** | Defines whether or not a file receives a time stamp in its name when it is received. Possible values:<br>`N` (default) Time stamp is added only if the file name already exists;<br>`Y` File name always receives the TransmissionID as time stamp. |
| **TraceItem** | Parameter that enables tracing. The following values are possible:<br>`O` (for Odette level),<br>`L` (for line level) and<br>`C` (for controller).<br>Trace output is written to the `$RVS_HOME/log/trace.log` file. |
| **TransmissionFailWaitTime** | Time in milliseconds for a transmission restart after a failure. |

**Example:**

### Excerpt from `rvsConfig.xml` ##################

```
...
<Environment>
<DB>c:\Programs\rvsEVO\jobs</DB>
<TEMP>c:\Programs\rvsEVO\files\temp</TEMP>
<INBOX>c:\Programs\rvsEVO\files\inbox</INBOX>
<OUTBOX>c:\Programs\rvsEVO\files\outbox</OUTBOX>
<ARCDIR>c:\Programs\rvsEVO\archive</ARCDIR>
<JobstartConfigFile>rvsJobstart.xml</JobstartConfigFile>
<StationsConfigFile>rvsStationlist.xml</StationsConfigFile>
<MonlogStylesheet>MonlogStylesheet.xslt</MonlogStylesheet>
<PersistenceArchive>RevisionLog.xml</PersistenceArchive>
<HostAllowFile>host.allow</HostAllowFile>
<HostDenyFile>host.deny</HostDenyFile>
<RMIServiceName>rvsEVO</RMIServiceName>
<RMIServiceHost>localhost</RMIServiceHost>
<JobQueueTimeout>3000</JobQueueTimeout>
</Environment>
...
```

########## `rvsConfig.xml` ##################

You can define and edit the paths for the `DB`, `TEMP`, `INBOX`, `OUTBOX`, `LOGDIR`, and `ARCDIR` elements.

**Example:**

```
<DB>D:\rvsEVO\jobs</DB>
```

```
<TEMP>C:\Programs\rvsEVO\temp</TEMP>
```

You are free to choose the names for the `rvsJobstart` and `rvsStationlist` files; the only requirement is that they are specified in the respective XML element, are valid XML files and are located in the conf directory. The same also applies to `HostAllowFile` and `HostDenyFile`.

**Example:**

```
<StationsConfigFile>stations.xml</StationsConfigFile>
```

The `stations.xml` file is a station list in XML format containing the required rvsEVO parameters (see chapter 3.2) and is located in the rvsEVO `conf` directory.
The entries for `RMISeviceName` and `RMIServiceHost` are for internal communication and must not be changed.

**Note**: You must stop and restart rvsEVO each time you have made changes to any configuration file (see chapter 2.4 and 2.5).

## 3.2 Customizing the station configuration

Station Configuration

The installer offers you the possibilty to set up a local station and the neighbour station.

For further customization of stations (creation of new stations and their modification or deletion) you can use:

– the Graphical User Interface (GUI) or

– configuration files in XML format.

The obligatory parameters for these two stations (the local and the neighbour station) have already been polled during installation, and the

```
$RVS_HOME\conf\rvsStationlist.xml
```

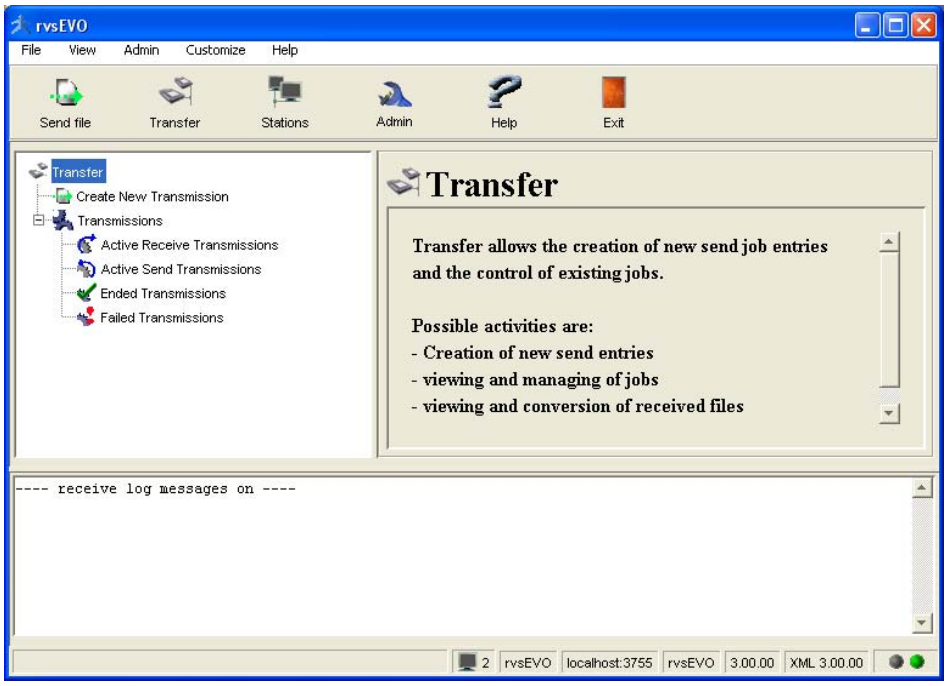
station configuration file has been appropriately adapted. This configuration is also visible in the GUI. If you do not need more entries for other stations, this completes the station configuration; sending and receiving files is now possible.

### 3.2.1 Customizing stations via GUI

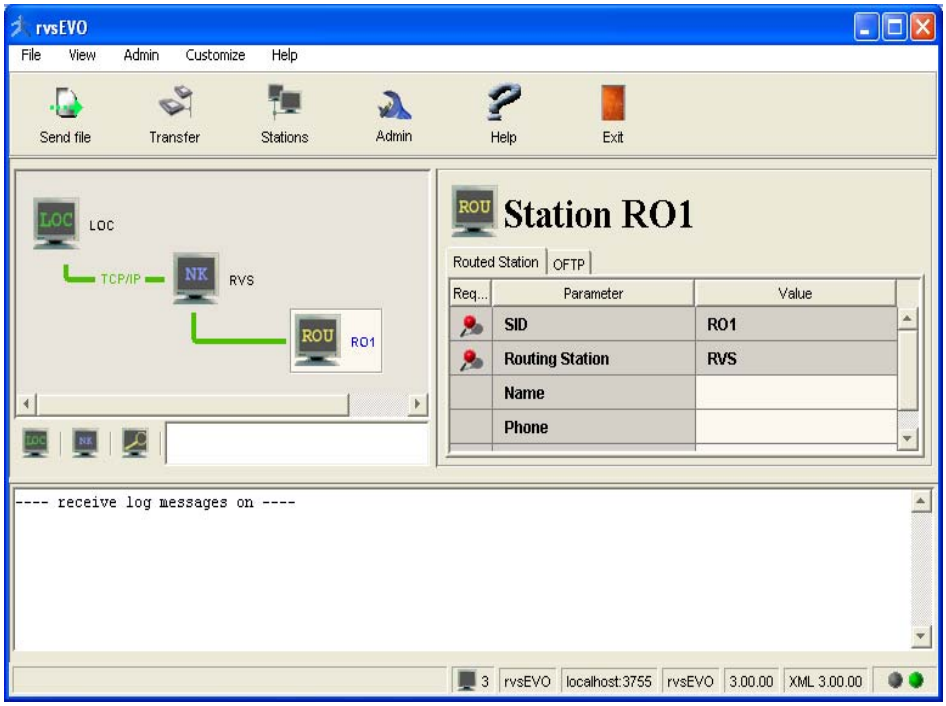
**Windows**: Start rvsEVO by choosing `Start -> Programs -> rvsEVO -> startGUI` (or the Start menu command you specified during installation) in the Start menu.

**Unix**: Start rvsEVO by starting the shell script `$RVS_HOME\bin\startGUI.sh`.

After the successfull start of rvsEVO the following window appears.




To open the station window, select the Stations icon in the function bar.




On the left side of the stations window you will see the station tree; to the right is a station parameter table.

The station tree depicts all of the stations which exist in the rvsEVO database (e.g. your local station and the neighbour station `RVS`, you have already configured during the installation, so they are now visible in the station tree) as well as their connection type (TCP/IP at present).

The station table on the right-hand side of the window displays all of the parameters for the station currently selected. With the aid of the various station tabs (OFTP, TCP/IP) you can configure various parameter groups.

Grayed fields indicate that these parameters cannot be edited.

**Note**: Once you have saved the parameters for a station, you can no longer change the station ID (SID). Changing the station ID would mean configuring an entirely new station.

The parameters which are obligatory for station configuration are marked in the column **Re**.. (Required) with the symbol.

**Example**: `Odette Id` is obligatory in the OFTP tab.

On the right-hand side of the window beneath the station parameter table is a series of buttons: Save, Cancel, Undo, Undo All. These allow you to save changes (Save), discard them (Cancel) or reverse them (Undo, Undo All).

### Configurating a local station

The possible tabs for the local station configuration are: Local Station, OFTP (Odette parameters), TCP/IP. Please read the chapter 3.2.3 and 3.2.4 for detailed explanation of those groups of parameters. The parameters `Name`, `Phone` and `Remarks` from the tab Local Station are comments. The tab TCP/IP is equivalent to the TCPIP_BASICS element (parameter group) in the XML station configuration file.

### Configuration of the neighbour station

The possible tabs for the neighbour station are: Neigbour Station, OFTP (Odette parameters), Line Type, TCP/IP. Please read the chapter 3.2.3 and 3.2.5 and for detailed explanation of those groups of parameters.

A right-click on the neighbour station opens the context menu, which offers then the options **Add routed station** and **Activate connection**. With the option **Add routed station** you can add a set of routed stations, that are reachable via the neighbour station . **Activate connection** activates the connection to the neighbour station.

### Setting up a routed station

Pre-condition: You must already have set up a direct neighbouring node via which you can reach the routed station.

Set up a routed station by a right-click on the neighbour station and selecting from the context menu the item **Add routed station**. For you, the connection type (line type) by which this station is to be reached is of no importance (this is dealt with by the direct neightbouring node. For this reason, the connection type to the routed station is not shown.

The possible tabs for this type of station are: Routed Station, OFTP (Odette parameters). Please read the chapter 3.2.3 and 3.2.6 for detailed explanation of those groups of parameters.

A routed station in rvsEVO can only be reached via the neighbour station. This is shown by the parameter Routing Station in the GUI that is equivalent to the parameter GATEWAY_STATION_NK in the element (parameter group) STATION_RT of the XML station configuration file.

Besides entering the freely selectable station name (SID) in the Routed Station tab, you only have to enter the `Odette ID` in the Odette tab to complete the station configuration of the routed station.

Once you have saved the parameters for a routed station, you can no longer change the station ID (SID). Changing the station ID would mean configuring an entirely new station without deleting the old one.

A right-click on the routed station opens the context menu, which offers then an option **Delete station**.

### 3.2.2 Customizing stations in the XML configuration file

On the other hand the station configuration can be done editing the XML station configuration file `rvsStationlist.xml` (see chapter 3.1.2 for the explanation, where this file is to be found). The STATION_LOC element in the configuration file is equivalent to the Local Station in the GUI, STATION_NK is Neighbour Station and STATION_RT is Routed Station in the GUI.

Excerpt from the `rvsStationlist.xml` file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<RVS_STATION_CONFIG>
<!-- Setting for the local station -->
<STATION_LOC>
 . . .
</STATION_LOC>
<!-- Setting for the neighbour station -->
<STATION_NK>
 . . .
<STATION_NK>
<!-- Setting for routed stations -->
<!-- One entry for each routed station -->
<STATION_RT>
. . .
</STATION_RT>
</RVS_STATION_CONFIG>
```

Comments

Lines starting with (<!--) and ending with (-->) are interpreted as comments.

Edit this file if you have to change mandatory parameters (**ODETTE_ID**, **TCPIP_BASIC**) or if you wish to assign values to other optional parameters such as **CONTACT**.

Use a text editor (e.g. `Edit`, `TextPad`) to do so. Please make sure to save your XML files as valid XML documents after editing as rvsEVO otherwise will not be able to read them and may fail to start correctly.

The changes in the GUI will be visible immediately (afer saving) in the XML station configuration file; the changes in the XML station configuration file will be visible in the GUI only after a new start of the rvsEVO (command `startGUI`).

**Note**: For correct TCP/IP communication you must ensure that the IP ports for RMI (1099) and for Odette (e.g. `3305`) are free.

The following chapters will describe all station parameters in detail.

### 3.2.3 Identification of rvsEVO stations

Station list

There are several parameters in the station list that identify rvsEVO stations. Of these only the **SID**, **ODETTE_ID** and **TCPIP_BASIC** parameters are obligatory. The SID parameter was assigned during rvsEVO installation; the parameter for the local station is `LOC`, and `RVS` for the a neigbour station.

| | |
|---|---|
| **SID** | This is a locally unique station ID which must consist of up to sixteen characters. It is a strictly local definition; remote stations do not have access to these names; they only know the ODETTE IDs. |
| **ODETTE_ID** | ODETTE ID is a worldwide unique identification of all nodes using the ODETTE file transfer protocol (OFTP). This 25 character name consists of<br>– the letter O,<br>– an 18 character organization identifier provided by the ODETTE codification group, and<br>– a 6 character computer sub address that is administrated by each organization.<br>If you communicate within your own closed network only, the ODETTE ID may be freely chosen as long as it remains unique in your network. |
| **CONTACT** | Contact information to be further specified later; these parameters are optional. |
| **../NAME** | Name of the contact partner responsible for this station (computer). |
| **../ENTERPRISE** | Company/enterprise |
| **../LOCATION** | Location |
| **../STREET** | Street |
| **../STREETNUMBER** | Street number |
| **../DEPARTMENT** | Department |
| **../PHONE** | Phone number |
| **../FAX** | Fax number |
| **../MAIL** | Email address |
| **../INET** | Enterprise website |
| **../FREETEXT** | At your disposal. |
| **TCPIP_BASIC** | TCP/IP parameters required for connection establishment. |
| **../IP_ADDR** | IP address or DNS name of the own or the partner station. |
| **../PORT** | Port on which a TCP/IP listener is to be started; 3305 by default. |
| **../SND_PW** | The password rvsEVO sends to the neighbor station. ODETTE password exchange between two neighbor stations and verification always occurs while a session is being established. |
| **../REC_PW** | The password rvsEVO expects from the neighbor station. |

| | |
|---|---|
| **../EERP_OUT** | Procedure for sending receipts (EERPs).<br>NORMAL: Generation of a receipt after successful file reception and immediate active transmission.<br>HOLD: Generation of a receipt after successful file reception. The receipt, however, is only sent after having been released with the handleEERP program. Default: NORMAL |

At present, the station list comprises the following, not yet evaluated parameters:

– The **<TCPIP_REC>** block with the **<MAX_INCOMING_SESSIONS>** and **<MAX_OUTGOING_SESSIONS>** parameters;
– The **ENABLED** parameter.

To apply for an ODETTE ID in Germany, please contact:

ODETTE ID

Verband der Automobilindustrie e.V. ( VDA )
Abt. Logistik
Postfach 17 05 63
60079 Frankfurt
Ph.: +49 69-7570-0

Get the complete description of OFTP from:

http://www.odette.org/

### 3.2.4 STATION_LOC station entry

Local station

This area comprises information on the local station; it must be present exactly once in the station configuration.

**Local station example:**

```
<STATION_LOC>
   <SID>LOC</SID>
   <ODETTE_ID>O0013005623GEDASSKK</ODETTE_ID>
   <CONTACT>
      <NAME>local station</NAME>
      <ENTERPRISE/>
      <LOCATION/>
      <STREET/>
      <STREET_NUMBER/>
      <DEPARTMENT/>
      <PHONE> </PHONE>
      <FAX/>
      <MAIL/>
      <INET/>
      <FREETEXT></FREETEXT>
   </CONTACT>
   <TCPIP_BASIC>
      <IP_ADDR>139.1.87.68</IP_ADDR>
      <PORT>3305</PORT>
   </TCPIP_BASIC>
   <TCPIP_REC>
      <MAX_INCOMING_SESSIONS>5</MAX_INCOMING_SESSIONS>
      <MAX_INCOMING_SESSIONS>1</MAX_INCOMING_SESSIONS>
```

```
   </TCPIP_REC>
   <ENABLED>Yes</ENABLED>
</STATION_LOC>
```

In the following example, the local station **SID** is `LOC`, the **ODETTE ID** `O0013005623GEDASSKK`. The neighbor node can reach this station under IP address `139.1.87.68` and at port `3305`.

**TCPIP_REC**

**Note**: The **TCPIP_REC** block with the **MAX_INCOMING_SESSIONS** and **MAX_OUTGOING_SESSIONS** parameters and the **ENABLED** parameter are not being evaluated at present and are reserved for future application. Do not edit nor delete these parameters.

### 3.2.5 STATION_NK station entry

neighbour station

This area contains information on a direct neighbor station. This area must be present in the station configuration for each neighbour station exactly once only. Each neighbour station has a different SID.

```
<STATION_NK>
   <SID>RVS</SID>
   <ODETTE_ID>O0013005623GEDASMEL</ODETTE_ID>
   <CONTACT>
      <NAME>neighbour station</NAME>
      <ENTERPRISE/>
      <LOCATION/>
      <STREET/>
      <STREET_NUMBER/>
      <DEPARTMENT/>
      <PHONE> </PHONE>
      <FAX/>
      <MAIL/>
      <INET/>
      <FREETEXT></FREETEXT>
   </CONTACT>
   <LINE_TYPE>
      <LINE_SUSP>No</LINE_SUSP>
      <PSESSIONS>-1</PSESSIONS>
      <ACTIVE_CON_SETUP>Yes</ACTIVE_CON_SETUP>
      <DELAY>0</DELAY>
   </LINETYPE>
   <ODETTE>
      <SND_PW>H5C</SND_PW>
      <REC_PW>L3Y</REC_PW>
      <SND_BLOCKS>0</SND_BLOCKS>
      <REC_BLOCKS>0</REC_BLOCKS>
      <EX_BUF_CRE>0</EX_BUF_CRE>
      <EX_BUF_SIZ>0</EX_BUF_SIZ>
      <EERP_IN>NORMAL</EERP_OUT>
      <EERP_OUT>NORMAL</EERP_OUT>
   </ODETTE>
   <TCPIP_BASIC>
      <IP_ADDR>139.1.65.42</IP_ADDR>
      <PORT>3305</PORT>
   </TCPIP_BASIC>
</STATION_NK>
```

| | |
|---|---|
| **SID** | Station ID (see chapter 3.2.3). |
| **ODETTE_ID** | ODETTE ID (see chapter 3.2.3). |
| **CONTACT** | Contact information (see chapter 3.2.3). |
| **LINE_TYPE** | For future application, not being evaluated. |
| **ODETTE** | Parameter block, comprises parameters for OFTP. |
| **../SND_PW** | The password rvsEVO sends to the neighbor. ODETTE password exchange between two neighbor stations and verification always occurs while a session is being established. |
| **../REC_PW** | The password rvsEVO expects from the neighbor. |
| **../EX_BUF_CRE** | For future application, not being evaluated. |
| **../EX_BUF_SIZ** | For future application, not being evaluated. |
| **../EERP_OUT** | Procedure for sending receipts (EERPs).<br>`NORMAL`: Generation of a receipt after successful file reception and immediate active transmission.<br>`HOLD`: Generation of a receipt after successful file reception. The receipt, however, is only sent after having been released with the `handleEERP` program. Default: `NORMAL` |
| **TCPIP_BASIC** | TCP/IP parameters required for connection establishment. |
| **../IP_ADDR** | IP address or DNS name of the own or the partner station. |
| **../PORT** | Port on which a TCP/IP listener is to be started; 3305 by default. |

### 3.2.6 STATION_RT station entry

Routing station

This entry defines for each routing station through which neighbors it can be reached.

```
<STATION_RT>
   <SID>ROUTEDSTATION</SID>
   <ODETTE_ID>OROUT</ODETTE_ID>
   <GATEWAY_STATION_NK>RVS</GATEWAY_STATION_NK>
   <CONTACT>
      <NAME/>
      <ENTERPRISE/>
      <LOCATION/>
      <STREET/>
      <STREET_NUMBER/>
      <DEPARTMENT/>
      <PHONE/>
      <FAX/>
      <MAIL/>
      <INET/>
      <FREETEXT/>
   </CONTACT>
```

```
   <ODETTE>
      <EERP_OUT>NORMAL</EERP_OUT>
   </ODETTE>
</STATION_RT>
```

| | |
|---|---|
| **SID** | Station ID (see chapter 3.2.3). |
| **ODETTE_ID** | ODETTE ID (see chapter 3.2.3). |
| **CONTACT** | Contact information (see chapter 3.2.3). |
| **GATEWAY_STATION_NK** | This field comprises the station's SID used to execute communication with the routed station configured here. For rvsEVO is may only be the direct neighbor station (rvs® center) because rvsEVO only allows for a direct neighbor station. |
| **ODETTE** | Parameter block, comprises parameters for OFTP. |
| **../EERP_OUT** | Procedure for sending receipts (EERPs).<br>`NORMAL`: Generation of a receipt after successful file reception and immediate active transmission.<br>`HOLD`: Generation of a receipt after successful file reception. The receipt, however, is only sent after having been released with the `handleEERP` program. Default: `NORMAL` |
| **TCPIP_BASIC** | TCP/IP parameters required for connection establishment. |
| **../IP_ADDR** | IP address or DNS name of the own or the partner station. |
| **../PORT** | Port on which a TCP/IP listener is to be started; 3305 by default. |

## 3.3 Customizing the JobStarts

The JobStart configuration comprises rules that allow special programs to be launched when appropriate files are being sent or received.

JobFilter

All programs of the jobFilters in question are started (their sequence cannot be influenced) if more than one jobFilter applies to the send or receive job.

It is possible to customize JobStarts via the GUI or via the JobStart configuration file.

### Customizing via GUI

At first you should open the Administration window selecting the Admin icon in the function bar. Then select the item Jobstart in the Administration tree on the left size. How to start GUI read please in the chapter 2.4.

It is possible to chosse between Jobstarts in receive direction and Jobstarts in send direction.  A Jobstart in receive direction is equivalent to

a resident receive entry in rvs®. A Jobstart in send direction is equivalent to a Jobstart after send attempt in rvs®.

A new Jobstart will be created with a right-click on a Jobstart after receive or a Jobstart after send attempt in the Administration tree (**Add new entry**). To select the already existing Jobstart, double-click the appropriate line of the appropriate JobStart in the right-hand window.

The following JobFilters' entries are possible: Direction, SID, VDSN, Process and SendAttempts. Please refer to the table of the JobFilters for a detailed description.

### Customizing via XML configuration file `rvsJobstart.xml`

Like most of the other rvsEVO configuration files this file is in the XML format as well.

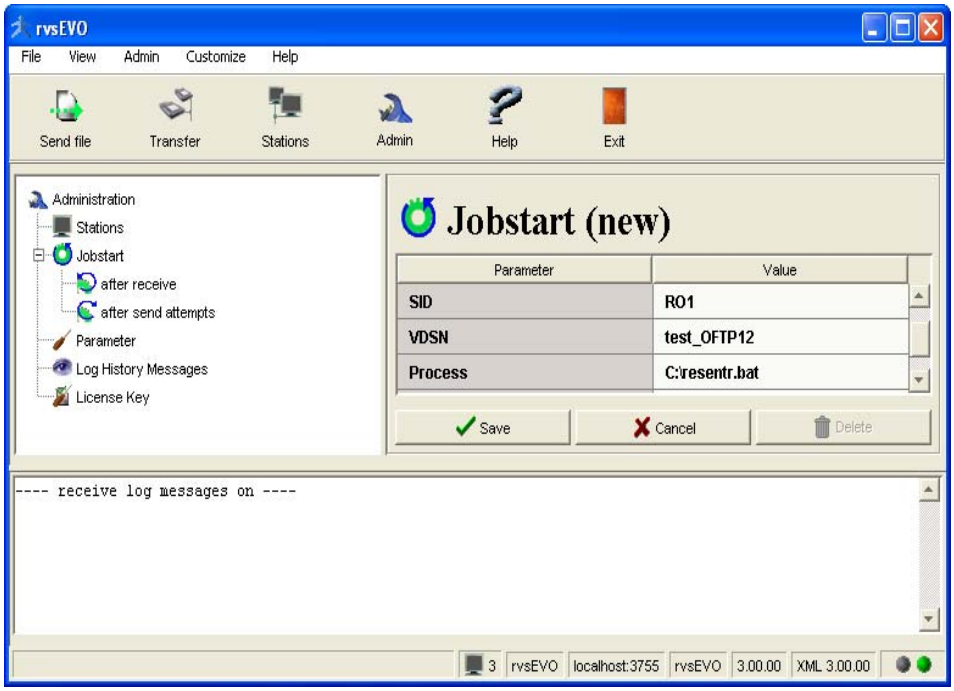
```
<jobstarterData>
       <jobfilters>
              <jobfilter>
                     <vdsn></vdsn>
                     <sid></sid>
                     <direction>SND</direction>
                     <sendAttempts>0</sendAttempts>
                     <process>C:\jobstart.bat</process>
              </jobfilter>
              <jobfilter>
               ...
              </jobfilter>
              ...
        </jobfilters>
</jobstarterData>
```

This file comprises any number of jobFilter elements. The table below gives a detailed description of individual jobFilter sub elements.

| | |
|---|---|
| vdsn | Virtual file name (regular expression as filter). |
| sid | Station ID (regular expression as filter for the station ID). |
| direction | Defines filter rules for the communication direction. Possible values: `SND` (when sending files), `RCV` (when receiving files). |
| sendAttempts | Number of failed attempts to send. Successful file transmission is indicated by "0" here. |

| | |
|---|---|
| process (mandatory) | Program to be started when all filter conditions apply. A defined set of parameters is passed to the programs.<br>Parameters:<br>1. jobId<br>2. Station ID (of sender or recipient)<br>3. File name of the file sent or received<br>4. VDSN<br>5. Date and time of the job<br>6. Number of attempts to send. |
| Shell | Command Shell for executions of the program (cmd for Windows; ksh, csh, ... on Unix systems |

Example (GUI) If rvsEVO receives a file with the virtual file name `test_OFTP12` from a routed station `RO1` the program `C:\resentr.bat` will be started.




Example (GUI) If rvsEVO sends successfully (`SendAttempts=0`) a file with the virtual file name `test-888` to the neighbour station `RVS`, the program `C:\sendjob.bat` will be started.


Delete a JobStart

If you want to delete a JobStart, select it at first with a double-click. Now you can remove it with the **Delete** button.

# 4 Working with rvsEVO

Batch files

The present chapter describes all programs available for everyday use of rvsEVO. These programs are located as batch files in the `$RVS_HOME\bin` directory or available via the rvsEVO GUI.

**Note**: To launch an rvsEVO program you must change to the `$RVS_HOME\bin` directory.

## 4.1 Starting the rvsEVO server

`startServer`

Use the `startServer` program to start rvsEVO Server. rvsEVO Server will be also started by the program `startGUI`. `startGUI` starts GUI and then the rvsEVO server (please read the chapter 2.4).

**Example:**

```
startServer
```

It is not possible to specify any parameters.

A successful start is indicated as follows:

```
*
* rvs Server has started.
*
```

**Note**: The RMI port 1099 is necessary for the RMI registry, so if this port is occupied, rvsEVO Server will not start successfully.

## 4.2 Stopping the rvsEVO server

`stopServer`

Use the `stopServer` program to stop rvsEVO.

**Usage:**

```
stopServer -m <mode> [-verbose]
```

All parameters are optional.

| | |
|---|---|
| **-m <mode>** | Time for jobs to terminate before rvsEVO is stopped.<br>Possible values:<br>`0` (default; 120 seconds),<br>`1` (60 seconds), 2 (30 seconds),<br>`3` (20 seconds),<br>`4` (10 seconds), |
| **-verbose** | Verbose message output. |

| | |
|---|---|
| **-help** | Requests help information. |
| **-?** | Requests help information. |

**Example:**

```
stopServer
```

**Result**: The server stops after 120 seconds.

**Example:**

```
stopServer -m 3
```

**Result**: The server stops after 20 seconds.

```
*
* rvs Server has stopped.
*
```

### 4.3 Displaying Monitor messages

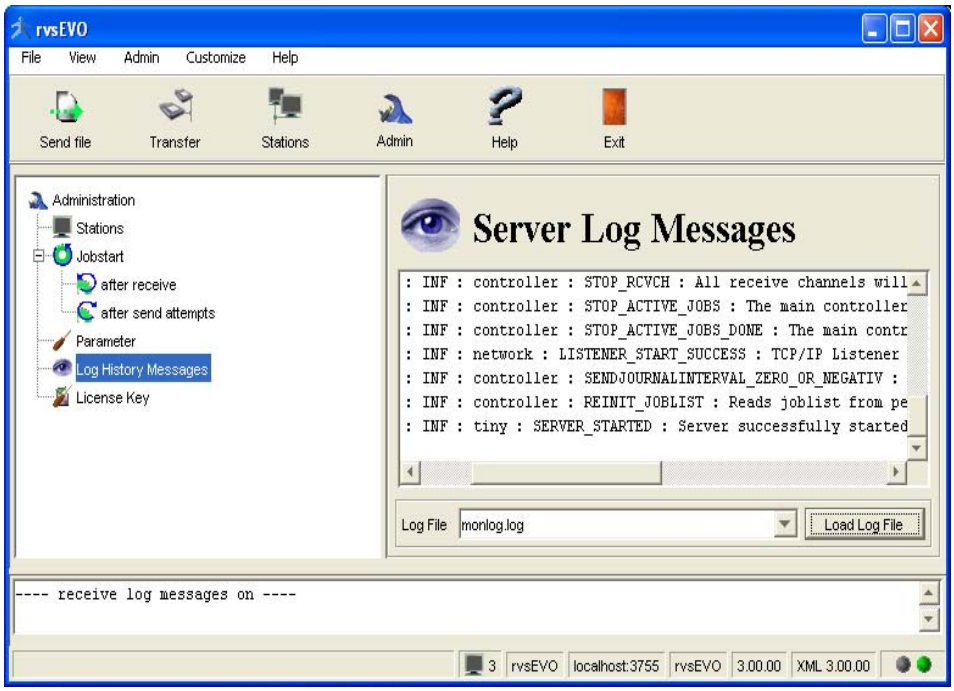
Log Messages are normally displayed in the bottom of the GUI window.

You can view old Monitor messages via the GUI or via the command line. Use this program to trace the current Monitor messages and to analyze error messages if applicable.

**GUI**

Select the Admin icon in the function bar of rvsEVO GUI . The Administration window opens with the Admin tree and the sub-entry Log History Messages. How to start of rvsEVO GUI please read the chapter 2.4.

Placing a tick in the box marked **Load log file** allows you to view the messages from the `$RVS_HOME\log\monlog.log` file. Only messages occuring after the GUI is started will be displayed.




**Command Line**

Use the `showMonitorLog` program to trace the current Monitor messages and to analyze error messages if applicable.

**Usage:**

```
showMonitorLog [-verbose]
```

Optional parameters:

| | |
|---|---|
| **-verbose** | Verbose message output. |
| **-help** | Displays a description of the current command |
| **-?** | Requests help information. |

The Monitor messages are written to the `$RVS_HOME \log\monlog.log` file.

### 4.4 Activating a station

`activate Station`

Use the `activateStation` program to activate the neighbour station in the command line. How to activate the neighbour station via the GUI, please see chapter 3.2.1.

**Note**: You cannot activate a routed station. You can activate only a direct neighbouring node (neighbour station).

**Usage:**

```
activateStation [-verbose]
```

Optional parameters:

| | |
|---|---|
| **-verbose** | Verbose message output. |
| **-help** | Displays a description of the current command. |
| **-?** | Requests help information. |

Heed messages in the command prompt window starting with message or error if a station activation fails (e.g. due to an incorrect IP address).

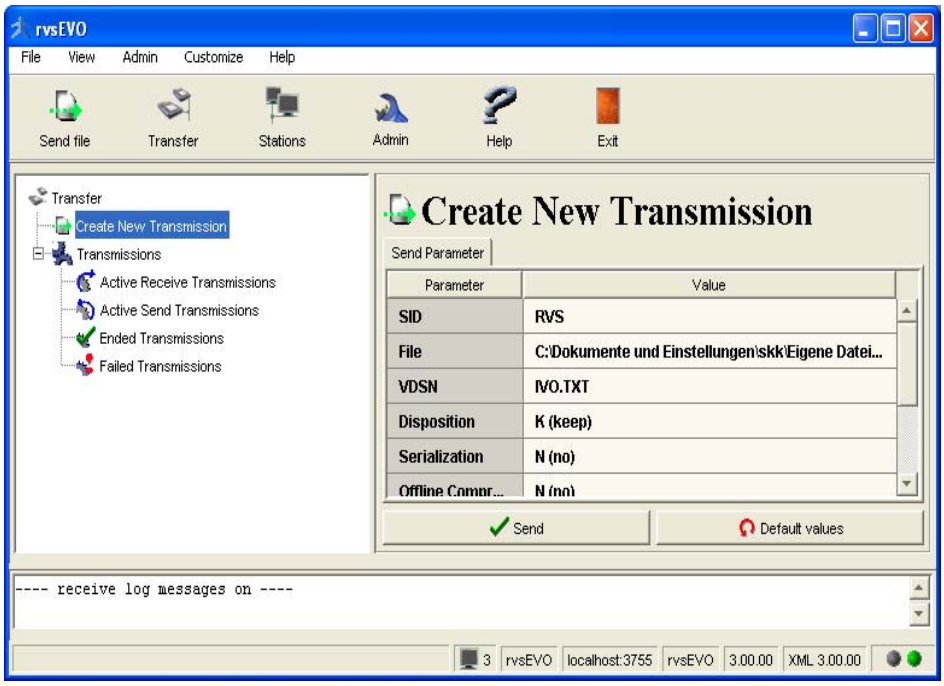
Correct the configuration file with the station table when the IP address is incorrect. You must stop and restart rvsEVO each time you have saved the configuration file.

### 4.5 Sending a file

It is possible to send a file via the GUI or via the command line. How to start GUI, please read the chapter 2.4.

**GUI**

Select the icon **Send file** or **Transfer** in the function bar. The Transfer window opens with the sub-entry Create New Transmission in the Transfer tree. Selection of Create New Transmission opens a window on the right-hand size of the screen entitled Create New Transmission. Now you can type or select the send parameters.




Required parameters:

**SID** Recipient station ID.

**File** File name of the file to be sent.

**VDSN** Virtual file name; the length of the file name used for ODETTE transfer must not exceed 26 characters.

Optional send parameters:

**Disposition**

- `K` (KEEP) file will not be deleted after sending
- `D` (DELETE) file will be deleted after sending

| | |
|---|---|
| **Format** | Format of the file to be sent:<br>– T=text file; a stream of ASCII characters<br>– U=unstructured (binary); byte stream<br>– V=variable; variable record length<br>– F=fixed; fixed record length<br>The files in format F or V must be text files with the appropriate line (record) length.<br>**Example**: If you want to send a file in format fixed with the record length 80, it must be a text file, where each line has a length 80 without CR/LF. The following parameters are to be set: Format=F, MaxRecl=80. |
| **Serialization** | Y (Yes)/N (No). If you set Serialization=Y, the files will be sent in the same order, as the send jobs were created. The next job will only be sent, if the previous is completely finished. All send jobs for the serialization must have the same LABEL. In the GUI the VDSN will be used as label. That means, that all jobs for the same serialization group must have the same virtual data set name (VDSN). |
| **Offline Compression** | – `Y` (Yes) ODETTE compression<br>– `N` (No) no ODETTE compression |
| **Encryption** | – `Y` (Yes) file will be sent encrypted<br>– `N` (No) file will be send without encryption |
| **MaxRecl** | maximal record length for the files in format F or V, please see the examle for the parameter Format. |
| **Conversion table** | For ASCII - EBCDIC conversion the following conversion tables are available : ASCII-IBM037, ASCII-IBM237, ANSI-IBM073, ANSI-IBM273.<br>For EBCDIC - ASCII conversion the following conversion tables are available : IBM037-ASCII, IBM237-ASCII, IBM037-ANSI, IBM273-ANSI.<br>Please see the note beneth the parameters' table for the command line for more explanation about the conversion tables. |
| **-h** | Requests help information. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

### Command Line

`createSendJob` Use the `createSendJob` program to send a file to the partner station via the command line.

**Usage:**

```
createSendJob -d <filename> -s <receiver sid>
[-I <input code> -O <output code>
 -t <conversion table>][-D <disposition>]
[-F <format>] [-M <length>] [-Y][-C]
[-S <serialisation> -l <label>][-h?][-verbose]
```

Required parameters:

**-d <filename>** File name of the file to be sent.

**-s <receiver sid>** Recipient station ID.

**-v <vdsn>** Virtual file name; the length of the file name used for ODETTE transfer must not exceed 26 characters.

Optional send parameters:

**-F <format>** Format of the file to be sent:

- T=text file; a stream of ASCII characters
- U=unstructured (binary); byte stream
- V (variable); variable record length
- F (fixed); fixed record length.

The files in format F or V must be text files with the appropriate line (record) length.

**Example**: If you want to send a file in format fixed with the record length 80, it must be a text file, where each line has a length 80 without CR/LF. The following parameters are to be set: `-F=F -M=80`

**-M <length>** maximal record length for the files in format F or V, please see the examle for the parameter Format.

**-I <input code>**

With the parameters input code and output code, you can select one of the conversion tables, that rvsEVO offers. The possible conversion tables are:

For ASCII - EBCDIC conversion the following conversion tables are available : ASCII-IBM037, ASCII-IBM237, ANSI-IBM073, ANSI-IBM273.

For EBCDIC - ASCII code conversion the following conversion tables are available : IBM037-ASCII, IBM237-ASCII, IBM037-ANSI, IBM273-ANSI.

Please, see the note beneath this table for details about code tables.

**Example**: If you want to use the conversion table ASCII-IBM237, as input code you should write ASCII (`-I ASCII`); as output code IBM237 (`-O IBM237)` .

It is also possibile to set as input/output code only the values `A`/`E`. In this case is `A` ASCII and `E` IBM037 (code conversion ASCII-IBM037).

**-O <output code>**

see input code.

**-t <conversion table>**

your own conversion table with the complete path. Please read the User Manual, chapter Code Conversion for more information how to create the own conversion table.

**-S <serialize>**

Y (Yes)/N (No). If you set the option `-S=Y`, the files will be sent in the same order as the send jobs were created. The next job will only be sent, if the previous is completely finished. All send jobs for the serialization must have the same LABEL. In the GUI the VDSN will be used as label. In the command line you can specify your own label.

**-l <label>**

Name of group of serialized send jobs. User specified (descriptive) label for this job. If you do not specify this parameter the VDSN will be user as a label.

**-D <disposition>**

– `K` (KEEP) file will not be deleted after sending

– `D` (DELETE) file will be deleted after sending

| | |
|---|---|
| **-C <compression>** | – `Y` (Yes) ODETTE compression |
| | – `N` (No) no ODETTE compression |
| **-Y <encryption>** | – `Y` (Yes) file will be sent encrypted |
| | – `N` (No) file will be send without encryption |
| **-h** | Requests help information. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

**Note**: The text file are stored on most systems in one of two computer codes, namely ASCII (American National Standard Code for Information Interchange) or EBCDIC (Extended Binary Coded Decimal Interchange Code).
ASCII is the standard code for UNIX and DOS/Windows systems. Here is the short expanation of the conversion tables, that are offered by rvsEVO.

EBCDIC was developed for IBM Mainframe computers.

- **ASCII**: US-ASCII ISO 646; the ASCII character set defines 128 characters (0 to 127 decimal). This character set is a subset of many other character sets with 256 characters, including the ANSI character set of MS Windows.
- **ANSI**: Windows ANSI, Values 0 to 127 are the same as in the ASCII character set, values 128 to 255 are similar to the ISO Latin-1 character set.
- **EBCDIC 037**: support characters, which are used in the following countries: Australien, Brasilien, Kanada, Neuseeland, Portugal, Südafrika, USA.
- **EBCDIC 273**: supports characters (especially umlauts), which are used in the following countries: Germany, Austria and Switzerland.

**Examples:**

```
createSendJob -d C:\text.txt -s RVS -v test
```

In this example the `C:\text.txt` file is sent to the station `RVS` with a virtual name `test33`. The virtual file name parameter (option `-v`) is mandatory.

**Examples:**

```
createSendJob -d C:\text.txt -s RVS -v OFTP_TEST
-F F -M 80 -I ANSI -O IBM273
```

In this example the `C:\text.txt` file is sent to the station `RVS` with a virtual name `OFTP_TEST`, this file is a text file of whicheach line has a length of 80 characters without CR/LF (`-F F -M 80`). Before transmission the file will be converted from ANSI to IBM037 (`-I ANSI -O IBM273`) code.

```
createSendJob -d C:\TEMP\part.txt -s RVS -v PART
-S Y -l AUTO
```

In this example the `C:\TEMP\part1.txt` file is sent to station `RVS` with a virtual name `PART`, this file belongs to the serialized group of files with the label (for the whole group) `AUTO`.

The message `createSendJob exited with return code 0` appears when a send job was successfully created.

A temporary file named, for example, `040329170027000` is created in the `SND` directory while a send job is being processed. This name is made up of the date (`040329`), the time (`170027`) and a consecutive three-digit number (`000`) for files arriving in the very same second.

In the following example job `040329170027000` waits for a receipt (**EERP**) from the station `RVS` and is therefore still in the `SND` directory. `SND` and `RCV` are directories where jobs currently being processed are intermediately stored. The Status field indicates the job's processing phase. `WF_EERP` (**W**aiting **F**or **E**ERP) means: waiting for EERP (receipt).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Job>
       <ID>040329170027000</ID>
       <Static>
              <FileName>C:\Answer.txt</FileName>
              <VDSN>TESTFILE</VDSN>
              <SID>RVS</SID>
              <Direction>SND</Direction>
              <Date>040329</Date>
              <Time>170027</Time>
       </Static>
       <Dynamic>
              <RestartPos>0</RestartPos>
              <RecCount>0</RecCount>
              <FilePos>19</FilePos>
                 <RcvBytes>0</RcvBytes>
                 <SendAttempts>0</SendAttempts>
                 <Status>WF_EERP</Status>
        </Dynamic>
</Job>
```

rvsEVO copies the job into the `ENDED` directory if it has been successfully terminated (e.g. after an `EERP` has been received). The `FAILED` directory holds jobs that could not be successfully processed.

Status Possible states for Send Jobs:

- RESTART=1 (wait after error to send file again).
- WF_SFID_ANSWER=2 (wait for answer for already sent SFID).
- WF_CDT=3 (wait for credit renewal).
- WF_EFID_ANSWER=4 (wait for answer to already sent EFID).
- WF_EERP=5 (wait for EERP).
- ENDED=6 (send job has ended).

- ENDED_WITH_JS_ERROR=7 (send job run into error during call of jobstart).
- FATAL_ERROR=8 (send job run into fatal error).

Possible states for Receive Jobs:

- RESTART (wait after error to receive file again).
- RESTART_AFTER_EFNA=2 (File could not be delivered; EFNA sent or not. Assuming partner will send file again).
- RESTART_AFTER_EFPA_FAILURE=3 (File already delivered, but EFPA failed. Assuming partner will send file again.).
- RECEIVING=4 (Receiving file data after EFPA).
- EERP_HOLDED=5 (File was completely received. EERP is on hold and needs to get released.).
- EERP_RELEASED=6 (User released EERP, but it is still not sent).
- EERP_DELETED=7 (User deleted EERP. Job will stop).
- ENDED=8 (Receive job has successfully ended after sending EERP.)
- ENDED_WITH_JS_ERROR=9 (Receive job run into error during calling jobstart).
- FATAL_ERROR=10 (Receive job run into fatal error).

**Note**: Please see in the rvs® portable, Reference Manual (chapter 3.3) for ODETTE protocol sequences (such as EFNA, SFID, EFID, EERP, ...).

### 4.6 Synchronization of Send Jobs

The Odette file transfer is asynchronous. That means: If you create a send job, a file will be only provided to be sent. The sending of the file will not possible, only after a connection to a partner station has been established.

In automated business with a lot of processes is desirable to react directly, if the sending of a file was not successfilly in a certain period of time.

As a solution for this problem rvsEVO offers on client-side a functionality for synchronized file transfer. This program remains active, until the transfer is successfully finished or an error occurs. It is possible to define the number of send attempts or the time in which a file fransfer has to be successfully done, or the whole transfer will be count as an error.

**Note**: A transfer is regarded as having been successfully completed when the ODETTE acknowledgement EERP (End-to-End Response) for this transfer has been received.

The `convertAndSend` program offers you the synchronization of send jobs functionality. It is similar to the `createSendJob program`: it has additionally the options for the synchronized file transfer and for the

conversion of an EDI message with the EDI converter WEDIConv (see WEDIConv User Manual).

**Usage:**

```
convertAndSend -d <filename> -s <receiver sid>
[-I <input code> -O <output code> -t <conversion
table>] [-F <format>] [-M <length>] [-S
<serialisation> -l <label>] [-za <attempts>] [-zt
<timeout>] [converter parms]

[-h?][-verbose]
```

Required parameters:

**-d <filename>** File name of the file to be sent.

**-s <receiver sid>** Recipient station ID.

**-v <vdsn>** Virtual file name; the length of the file name used for ODETTE transfer must not exceed 26 characters.

Optional send parameters:

**-F <format>** Format of the file to be sent:

- T=text file; a stream of ASCII characters
- U=unstructured (binary); byte stream
- V (variable); variable record length
- F (fixed); fixed record length.

The files in format F or V must be text files with the appropriate line (record) length.

**Example**: If you want to send a file in format fixed with the record length 80, it must be a text file, where each line has a length 80 without CR/LF. The following parameters are to be set: `-F=F -M=80`

**-M <length>** maximal record length for the files in format F or V, please see the examle for the parameter Format.

| | |
|---|---|
| **-I <input code>** | With the parameters input code and output code, you can select one of the conversion tables, that rvsEVO offers. The possible conversion tables are:<br><br>For ASCII - EBCDIC conversion the following conversion tables are available : ASCII-IBM037, ASCII-IBM237, ANSI-IBM073, ANSI-IBM273.<br><br>For EBCDIC - ASCII code conversion the following conversion tables are available : IBM037-ASCII, IBM237-ASCII, IBM037-ANSI, IBM273-ANSI.<br><br>Please, see the note beneath this table for details about code tables.<br><br>**Example**: If you want to use the conversion table ASCII-IBM237, as input code you should write ASCII (`-I ASCII`); as output code IBM237 (`-O IBM237)`.<br><br>It is also possibile to set as input/output code only the values `A`/`E`. In this case is `A` ASCII and `E` IBM037 (code conversion ASCII-IBM037). |
| **-O <output code>** | see input code. |
| **-t <conversion table>** | your own conversion table with the complete path. Please read the User Manual (chapter Code Conversion) for more information how to create the own conversion table. |
| **-S <serialize>** | Y (Yes)/N (No). If you set the option `-S=Y`, the files will be sent in the same order as the send jobs were created. The next job will only be sent, if the previous is completely finished. All send jobs for the serialization must have the same LABEL. In the GUI the VDSN will be used as label. In the command line you can specify your own label. |
| **-l <label>** | Name of group of serialized send jobs. User specified (descriptive) label for this job. If you do not specify this parameter the VDSN will be used as a label. |
| **-za <attempts>** | count of send attempts for synchronized transmission. |
| **-zt <timeout>** | time-out for synchronized transmission in seconds |

Optional convert parameters:

| | |
|---|---|
| **-cf** | format description for converter step 1 |
| **-cf2** | format description for converter step 2 (optional) |
| **-ct** | stylesheet for converter step 1, optional |
| **-cd** | converter direction for converter step 1, default: EDI2XML |
| **-cd2** | converter direction for converter step 2, default: XML2EDI. |
| **-cl** | log level for converter, default: 0 |
| **-cs** | line feed behaviour for the EDI-message output;<br>0 - no LF as segment separator,<br>1 - LF as segment separator (default). |
| **-cI** | identation for XML output;<br>0 - no identation; 1 - identation on (default). |
| **-ce** | encoding for the XML output; default: UTF-8. |

Optional parameter:

**-verbose**

**-help**

**-?**

**Examples:**

```
convertAndSend -d C:\teil56.txt -s RVS -v TEILE
-za 4
```

In this example the `C:\teil56.txt` file is sent to the station `RVS` with a virtual name `TEILE`, count of send attempts is limited to 4.

```
convertAndSend
-d C:\INTEGRATION\test.txt -s RVS -v TEST
-cf C:\rvsET\system\fmtDesc\fw.kanban.ineas.xml
-ct C:\rvsET\system\stylesheets\ineas2deljit.xslt
-cf2 C:\rvsET\system\fmtDesc\edifact.97.orig.xml
```

In this example the `C:\INTEGRATION\test.txt` file is sent to the station `RVS` with a virtual name `TEST`. The `test.txt` file was also converted with the EDI converter **WEDIConv** (installed in the directory `C:\rvsET\`); first from the inhouse format `test.txt` to the XML

format (`fw.kanban.ineas.xml`) and then to the EDIFACT message (`edifact.97.orig.xml`). In the first step was used a stylesheet `ineas2deljit.xslt` for the special presentation.

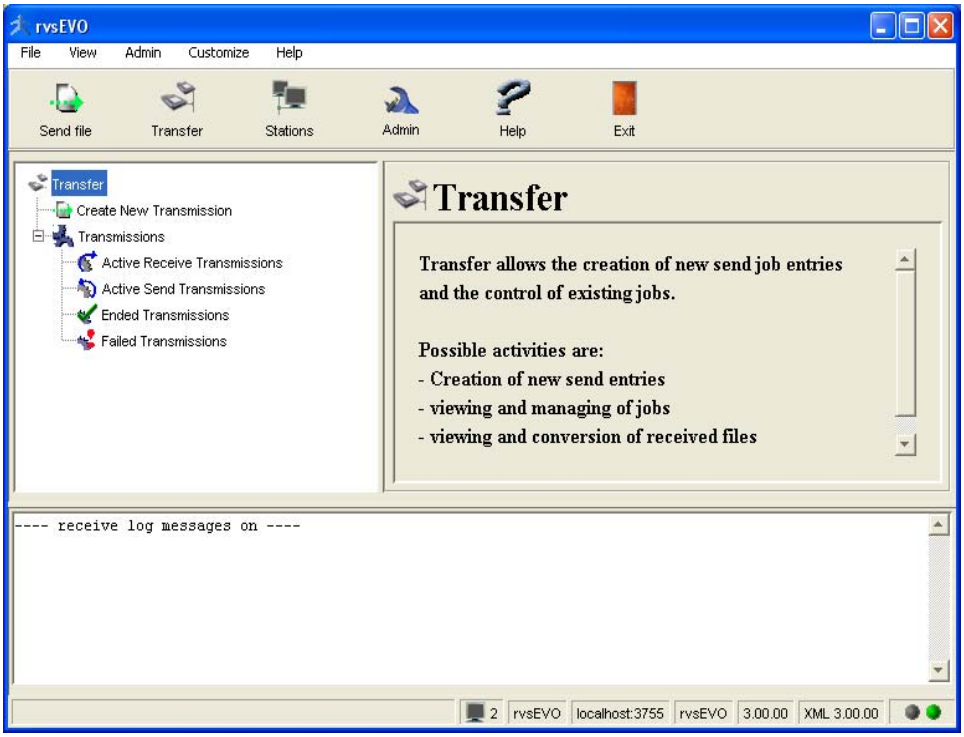
**Note**: If you use `convertAndSend` to send files, you must pay attention to the fact, that with this feature a new send job after an occured error can cause a double transmission (e.g. if the file is already transmitted, but the in the `-zt` parameter defined time out period passed without receiving the EERP. The transmission can be still active independent from the own local station. In this case, the new sending of the file can cause double transmission. The application above rvs® must be able to handle this particular situation. A possible solution is: the file name must be unique or it must get a unique counter (a counter stamp)).

### 4.7 Listing of all receive and send jobs

The most important information about the send and receive jobs can be shown via the GUI or via the command line. How to start rvsEVO, please read the chapter 2.4.

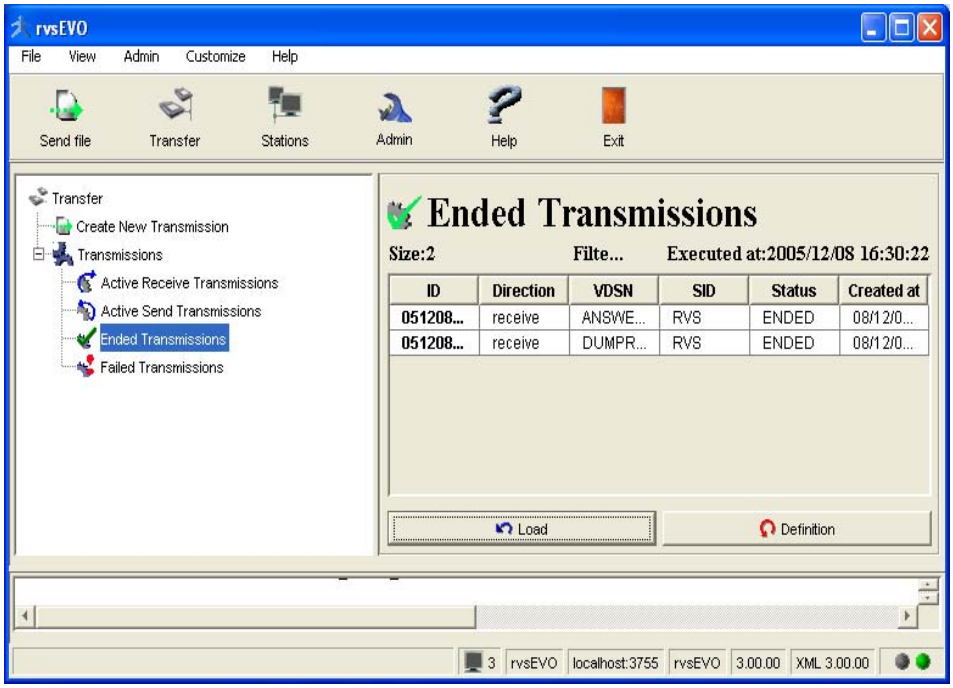
#### GUI

Select the icon Transfer in the function bar. The Transfer window opens with the sub-entry **Transmissions** in the Transfer tree. Now you can select between the Active Receive/Send Transmissions, Ended Transmission and Failed Transmissions.

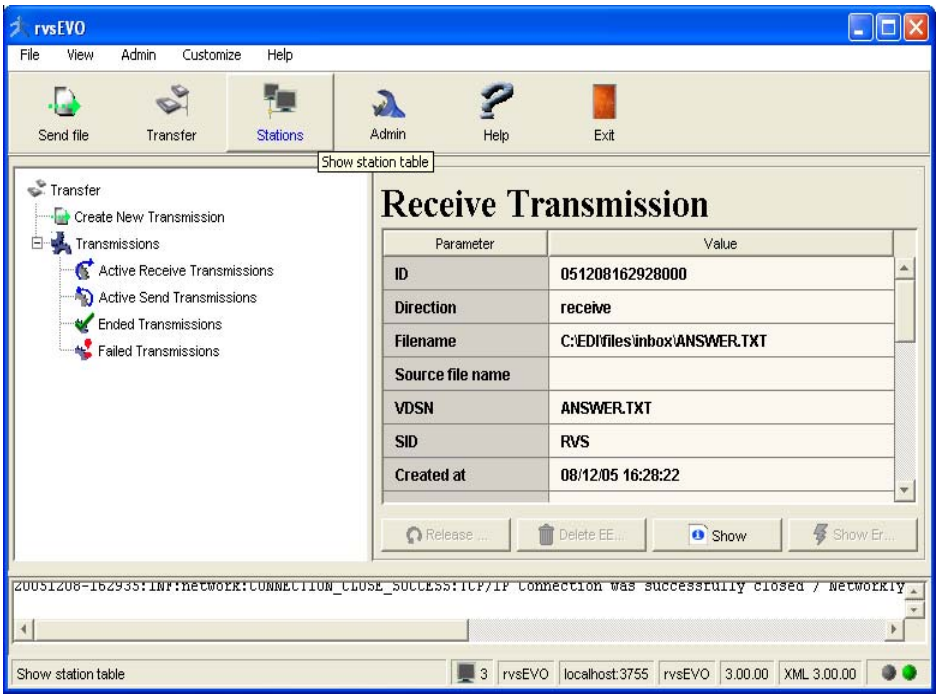


**Note**: A transfer is regarded as having been successfully completed when the ODETTE acknowledgement EERP (End-to-End Response) for this transfer has been received.

Single-click on selected job sub-ordner gives you an overview of all jobs in this ordner in the right-hand window.




By double-clicking on a job line in the right-hand section of the window you can obtain a detailed view of the relevant job.




Active Receive Transmissions: If you click on the line Active Receive Transmission and then double-click on a particular job line, you will see all details about the particular receive job in transmission.

Release EERP_OUT

With the buttons in this right-hand window you can change the status of the EERP_OUT for this particular job. The Button **Release** releases the EERP_OUT in status Hold (status=EERP_HOLDED).

Delete EERP_OUT

The Button **Delete** deletes the EERP_OUT. The same functionality is available by the program `handleEERP` (see chapter 4.8).

Delete/Release Job

Active Send Transmissions: To delete this job, activate the **Delete** button. If the job is active, you must first pause it, as rvsEVO will not allow you to delete a job which is actually being transmitted. To pause the job, use the button marked **Hold**. A job, which has been paused can be released again by pressing the **Release** button.

**Command Line**

`getJobList`

Use the `getJobList` program to list all jobs.

**Usage:**

```
getJobList [-a] [-e] [-f] [-verbose]
```

Optional parameters:

**-a** Detailed information on jobs currently being processed.

| | |
|---|---|
| **-e** | Information on terminated jobs. |
| **-f** | Information on failed jobs. |
| **-h** | Requests help information. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

**Example:**

```
getJobList -e
```

**Result:**

```
************************
job 051208162928000 (RCV): state=ENDED
job 051208162930000 (RCV): state=ENDED
************************
List ended. size=2
************************
```

**Information on a job entry**

`getJob` Use the `getJob` program to retrieve information on a particular send or receive job.

**Usage:**

```
getJob -n <jobid> [-a] [-verbose]
```

Required parameters:

| | |
|---|---|
| **-n <jobid>** | Information on a send or receive job with ID `<jobid>`. |

Optional parameters:

| | |
|---|---|
| **-a** | All available job information is given. |
| **-help** | Requests help information. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

**Example:**

```
getJob -n 040329173456000
```

**Result:** `job 040329173456000 (RCV): state: ENDED`

### 4.8 Deleting or releasing EERPs

`handleEERP` Use the `handleEERP` program to delete or release receipts (`EERP_OUT`). It is also possible to delete or release EERPs via the GUI (Please see the chapter 4.7).

**Usage:**

```
handleEERP -r|-d <num> [-verbose]
```

Required parameters:

| | |
|---|---|
| **-r \| -d <num>** | ID of the job for which the EERP is to be released/ deleted. |

Optional parameters:

| | |
|---|---|
| **-help** | Requests help information. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

## 4.9 Archiving the entries of processed send or receive jobs in the revision log

`archiveJobs` The `archiveJobs` program creates a `RevisionLog.xml` file in the `$RVS_HOME\archive` directory; this file is used to log successfully processed send or receive jobs.

**Usage:**

```
archiveJobs [-verbose] [-help] [-?]
```

Optional parameters:

| | |
|---|---|
| **-help** | Displays a description of the current command. |
| **-verbose** | Verbose message output. |
| **-?** | Requests help information. |

Terminated jobs are deleted from the `$RVS_HOME\jobs\ENDED` directory and the data is written into the `RevisionLog.xml` file in the `$RVS_HOME\archive` directory.

**Note**: If you have more as one `RevisionLog.xml` file, they would be differentiate by a timestamp.

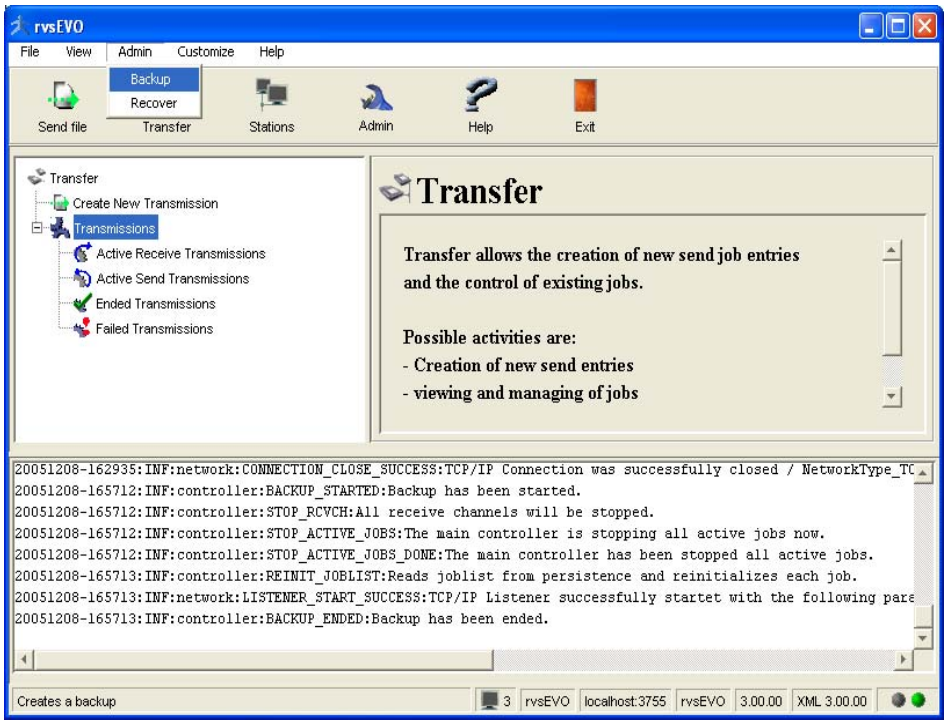
# 5 Backing Up and Recovering rvsEVO Data

rvsEVO lets you back up all relevant data and recover them if necessary. This is particularly important when an error has occurred in rvsEVO and the user wishes to revert to the old status prior to the error.

## 5.1 Back-up

Prerequisites: The back-up function only starts when the absence of communication has been assured (no file reception/transmission and no encryption/compression). The back-up script will terminate processes still running.

To perform the back-up, choose `Admin/Backup` menu item in the rvsEVO GUI window.




As an alternative you can also start this function by launching a script at the command prompt.

**Syntax:**

```
createBackup [-d <dir>] [-verbose][-help][-?]
```

All parameters are optional:

**-d <dir>** Specifies the back-up directory; without this option the back-up data will be written to the `$RVS_HOME/archive` directory.

| | |
|---|---|
| **-verbose** | Displays detailed messages. |
| **-help** | Requests help information. |
| **-?** | Requests help information. |

**Note**: You can use the new `BackupOnStartup` parameter to specify an automatic back-up to be performed each time you start rvsEVO. Set this parameter in the graphical user interface (Admin window) or in the `rvsConfig.xml` configuration file. Default value is `Y` (Yes).

### 5.1.1 What is backed up?

Back-up covers the following files or directories:

– The entire `$RVS_HOME/conf` directory containing the rvsEVO configuration files.

– Back-up of job files in the `$RVS_HOME/jobs/SND` and `$RVS_HOME/jobs/RCV` directories. This backs up temporary, not fully processed jobs.

– files from the `$RVS_HOME/system/data` directory.

The back-up data is written to a `<Back-upTime>.jar` file. `Back-upTime` is an automatically assigned file name, generated according to the YYMMDDHHMMSS format and featuring an additional 3-digit counter.

**Example**: The `051010112417000.jar` is the back-up file generated on 2005-10-10 at 11:24:17. At this time, there were no other back-up files, resulting in the additional counter value of `000`.

### 5.1.2 Redo Log

Starting at the back-up time, any dynamic data is logged in a continuous log (Redo Log).

“Dynamic data” refers to information on send and receive jobs. This data is written in order to be able to recover incomplete transmission jobs at a later time. Completed jobs are logged but are not relevant for recovering.

The Redo Log is assigned the same time stamp as the pertaining back-up file, bearing the `Redo_` prefix and the `.log` extension and can thus be clearly assigned to a back-up. Like the back-up file, this file is written to the `$RVS_HOME/archive` directory.

**Example**:

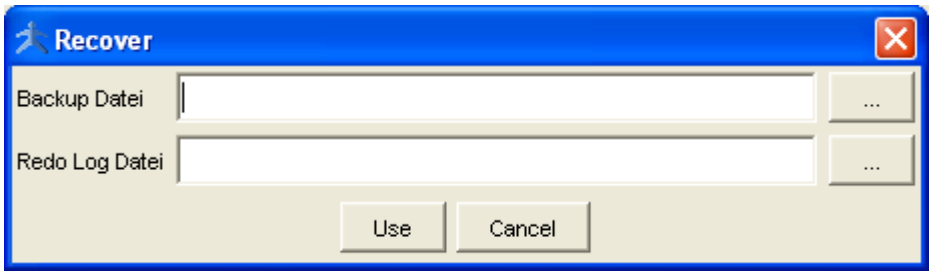
In addition to the `051010112417000.jar` back-up file the pertaining `Redo_051010112418000.log` Redo Log file was generated.

A pertaining Redo Log file is generated as soon as a new back-up was made. The `.log` extensions of all existing Redo Log files will be replaced by `.old`.

The Redo Log file is a text file. Each line comprises a `<Job>` XML element with all relevant job information such as `ID`, `FileName`, `VDSN`, `SID`.

### 5.2 Recovering the rvsEVO data

To recover rvsEVO data, choose the `Admin/Recovery` menu item in the rvsEVO GUI window. In the dialog that opens, you can specify the names for the back-up file and the Redo Log. Back-up file and Redo Log will be searched for in the `$RVS_HOME/archive` directory when no path has been specified.




As an alternative you can start this function by typing `$RVS_HOME/bin/doRecover.bat` at the command prompt.

**Syntax:**

```
doRecover -b <dir> -r <name> [-d <name>]
[-help][-?]
```

| | |
|---|---|
| **-d <dir>** | Mandatory parameters: Name of the back-up file; you can specify the complete path with directory or just the file name without directory. In the latter case the back-up file is expected in the `$RVS_HOME/archive` directory. |
| **-r** | Name of Redo Log file. You can specify the complete path with directory or just the file name without directory. In the latter case the Redo Log file is expected in the `$RVS_HOME/archive` directory. |
| **-verbose** | Displays detailed messages. |
| **-help** | Requests help information. |
| **-?** | Requests help information. |

**Note**: Any send/receive processes as well as the Service Provider (encryption/compression) will be terminated prior to recovery start.

Redo Log recovery is logged in the `$RVS_HOME/log/monlog.log` file.

# 6 Encrypted transmission with rvsEVO

The present chapter describes the basics of encrypted transmission with rvsEVO and key administration.

## 6.1 Introduction: basics

For encryption, rvsEVO uses the same component (Com-Secure) as rvs®. This combines the benefits of symmetrical and asymmetrical techniques: the high speed of the symmetrical and the security level of the asymmetrical technique. rvsEVO uses the following techniques:

– **3DES** as symmetrical technique (length: 3x56 bits = 168 bits),
– **RSA** as asymmetrical technique (length: 768 to 2048 bits),

Electronic signature

For increased security, the encryption component uses an electronic signature. The signature ensures that data do not undergo any unnoticed changes during transmission.

## 6.2 Principle and sequence of rvsEVO encryption

Creating a key pair

Each participant in encrypted communication locally creates a key pair, comprising the **public key** and the **private key**.

Distributing the public key / safely storing the private key

He provides the public key to each partner he expects files from. This allows the data to be exactly encrypted for the partner who sent this public key. You can safely distribute the public key since this key alone is not sufficient for decryption.

Each participant keeps his private key and stores it safely.

Three keys are required for decryption (the own key pair and the partner's public key). It will no longer be possible to decrypt files sent by the partner if one of the three necessary keys is lost.

## 6.3 How do I create an own key pair?

We use the `Portecle` Open Source tool (*http://portecle.sourceforge.net*) as graphical user interface for key administration. The *http://portecle.sourceforge.net/howtos.html* website offers user documentation in HTML format because this tool features many more functions than are necessary for rvsEVO. The present chapter will only cover those functions that are necessary for rvsEVO.

Call this feature with a rvsEVO program group: `Start -> Programs rvsEVO -> Key Management.`

You can create a new key pair with the menu item `Tools/Generate Key Pair`.

Administration of own (private and public) and of partner keys (only public) occurs in a keystore file.

You must set the path to the keystore file to be used in the `$RVS_HOME/conf/cryptoParameter.xml` configuration file (XML element: `keyStoreParameter`; subelement: `fileName`). The default is: `$RVS_HOME/system/data/keystore.p12`.

**Note**: The extension `.p12` indicates that this keystore file is in the PKCS #12 format.

**Creating a key pair**

Use `Tools -> Generate Key Pair` in the menu to create a new key pair.

Select `RSA` as `Key Algorithm` and the default of 1024 as `Key Size` in the **Generate Key Pair** window.

Select `SHA1withRSA` as `Signature Algorithm` in the **Generate Certificate** dialog that appears next. `Validity` indicates the number of days the keys are valid.

The `Common Name` parameter only applies in case of a connection to an existing PKI (Public Key Infrastructure) and if this parameter is required for the LDAP structure.

All other data concerns your organization. rvsEVO does not stipulate how to complete the fields.

Once you made all entries in the **Generate Certificate** window and clicked **OK** to confirm, the **Enter Alias Name** window appears where you (**IMPORTANT!!!**) must specify the station ID of the station for which the key pair is generated. The default is LOC, as this is your local station.

### 6.4 How do I create an Com-Secure format from an own public key?

You must export your own public key in the Com-Secure format before you can communicate with other partners.

Again, you will be using the `portecle` tool.

The following steps are necessary:

– Right-click on the key pair you wish to export the public key from.

– Choose the **Export** item in the context menu.

– Choose the `Head Certificate` option as `Export Type` and the `DER Encoded` option as export format in the **Export Keystore Entry** dialog.

– You should save the generated certificate with a file name bearing the `cer` extension.

– You can double-click this file in the Explorer to view it.

### 6.5 Ho do I import the partner’s public key into the keystore file?

You can import a partner’s public key in the Com-Secure format only.

For exchanging with the rvs® encryption component (Com-Secure) you must first convert the public key of the rvs® station using the `rvskeyreq` tool (see the rvsXP or rvsX user manual) into a certification request.

The certification request generated from the `request.txt` file using the `rvskeyreq` command must then be converted to a X.509 certificate with the `convertKey` rvsEVO tool before it can be imported into the `keystore` file.

**Syntax**:

```
convertKey pkcs10file
```

**Example**:

```
convertKey C:\request.txt
```

**Result:**

The result is the `request.txt.cer` file created in the same directory where you have launched the `convertKey` tool. This file has the extension `.cer` for certificate. To simplify matters you can rename this file to `request.cer`.

**Note**: Partners not using rvs® must send you their public key in the Com-Secure format.

As already mentioned earlier in this chapter, you can import your partner's public key into the keystore file in the Com-Secure format only.

The following steps are necessary:

– Open your keystore file with the `portecle` tool.

– `Tools -> Import Trusted Certificate`

– In the **Import Trusted Certificate** dialog you can browse to the file with the X.509 certificate and import it into the keystore file by pressing the **Import** button.

– The following dialogs prompt you to confirm that the certificate information is correct and that you accept this certificate.

– Specify the station ID of the partner for whom you import the certificate in the **Trusted Certificate Entry Alias** dialog.

Please refer to chapter 4.5 "Sending a file" on how to send an encrypted file with rvsEVO.

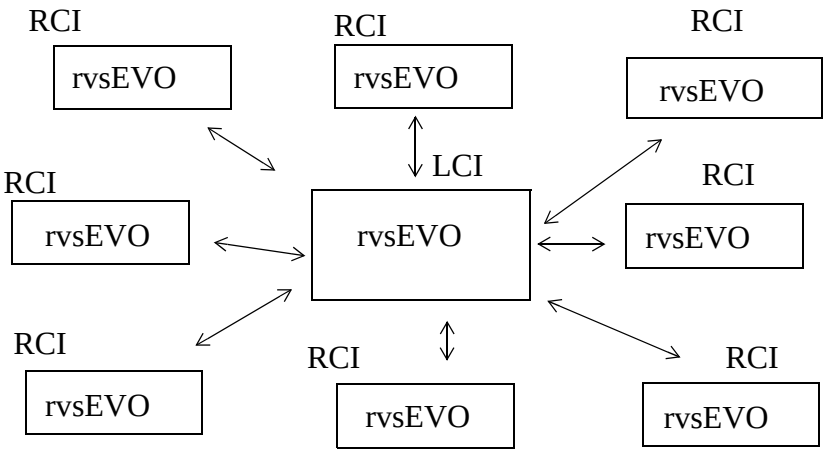
# 7 rvsEVO Central Administration

This chapter describes the powerful feature of rvsEVO: how to remotely administer other rvsEVO stations. It is essential for the network administrator being able to remotely administer all rvsEVO installations.

## 7.1 Introduction

The rvsEVO Central Administration enables the configuration of numerous rvsEVO installations by using special configuration files.

The configuration files will be sent from a rvsEVO station (we call it: Local Configuration Instance LCI) to the station, that should be administrated (we call it: Remote Controlled Instance RCI). The both instances are located in the rvsEVO star network (Please see the following picture).




To distinguish the configuration files for the central administration from the normal rvsEVO XML configuration files such as `rvsStationlist.xml`, the files for the central administration will be named **configuration container files**.

These are

– `cfg.req.jar` for configuration requests, sent by LCI to RCI and

– `cfg.rsp.jar` for configuration responses, sent by RCI to LCI.

### The Local Configuration Instance LCI

Each rvsEVO installation can be used as LCI (please see the chapter 2.3 for the explanation how to install rvsEVO).

The LCI maintains a special directory where the configuration data of all RCI's (rvsEVO stations, that are to be administrated) are stored - the Configuration Repository CRep. The location of CRep is `$RVS_HOME/`

management. Please refer to the chapter 1.2 for the explanation of $RVS_HOME.

The entries of CRep ($RVSTINY_HOME/management) are:

– $RVS_HOME/management/mgmt-datastore
  This directory will be created only after a succesfull transmission and response to the request for the configuration file of the RCI. Please see the chapter 7.2 for the explanation how to make a configuration request.
– $RVS_HOME/management/mgmt-log/activity-log This log file contains protocols of all configuration actions. Each entry comprises a timestamp, the type: **configuration request** or **configuration response** respectively, the SID of the administrated rvsEVO station (RCI) and a message text.
– $RVS_HOME/management/mgmt-templates
  This directory contains templates for management actions. It is not necessary to deal with this directory except for updates of the administration software itself.
– $RVS_HOME/management/mgmt-workspace
  This directory stores configurations of RCIs for editing. It contains subfolders, that are named after the RCI's SID (see the tool prepareUpdateStation, chapter 7.2 for an example).

#### The Remote Controlled Instance RCI

The RCI is a is rvsEVO station, which should be administrated remotely.

When rvsEVO (RCI) receives a configuration request (a file cfg.req.jar) a dedicated job will be launched. This job starts the rvsEVO configuration process that handles the configuration request and generates the configuration response. The configuration response is sent back from the RCI to LCI as file cfg.rsp.jar.

### 7.2 Command Tools of the Central Administration

The whole process (all configuration and administration cases) is to be done by the following command tools. These tools are located in the directory $RVS_HOME\bin as batch files (please see the chapter 1.2 for the explanation of $RVS_HOME.), so you must change to this directory to be able to execute them.

| | |
|---|---|
| orderConfiguration.bat -s SID | fetches the configuration of an RCI and stores it in CRep.<br><br>**Example**:<br>`C:\rvsEVO\management\mgmt-datastore\TINYPW`<br><br>In this example `TINYPW` is the SID of RCI (of the station, that should be administrated). |
| prepareUpdateStation.bat -s <SID> | gets a RCI configuration copy out of CRep to the WorkDir in order to serve as starting point for modifications.<br>**Example** for the WorkDir:<br><br>In the directory `C:\rvsEVO \management\mgmt-workspace` the following subdirectory will be created `TINYPW\UPDATE_STATION_040826_114418\out.`<br><br>This is a subdirectory for the `TINYPW` station; the date of the creation is 2004-08-26, the time of the creation is 11:44:18. |
| commitUpdateStation.bat -s <SID> -d <WorkDir> | sends a modified configuration from the WorkDir (without `out` subdirectory) to an RCI (RCI should be set with the option -s for stationID).<br><br>**Example**:<br><br>`commitUpdateStation.bat -s TINYPW -d C:\rvsEVO\management\mgmt-workspace\TINYPW\UPDATE_STATION_040826_114418` |

**Note**: The name of the WorkDir directory consists of CRep (see 7.1); RCI'SID (`TINYPW` in the table example), directory `UPDATE_STATION` with the timestamp (consisting of date and time; in the table example `040826_114418`) and the directory `out`.

WorkDir contains all essential rvsEVO directories and files: By changing these files the following configuration actions for example are possible:

– Modify station configuration by modifying `$RVS_HOME/conf/rvsStationlist.xml`.

– Modify jobstarts by modifying `$RVS_HOME/conf/rvsJobstart.xml`.

– Update the software by replacing .jar files in the directory `$RVS_HOME/lib`.

### 7.3 How to work with the central administration features?

The following steps will show you the typical configuration run. It may be used as a basic example when a rvsEVO network administrator deals with rvsEVO Central Administration. These steps are always necessary, independent of the fact, whether you want to make an update, change some rvsEVO parameter or exchange the license key file. More details about the particular administation tasks, you will find in the chapters 7.3.1, 7.3.2 and 7.3.3.

**Note**: The following commands are available as batch files in the `$RVS_HOME\bin` directory (please see the chapter 1.2 for the explanation of `$RVS_HOME`.), so you must change to this directory to be able to execute them.

- at first, get the configuration of the RCI (of the rvsEVO station, that should be administrated). Use the program `orderConfiguration` for this step.
  **Example**: If you would like to administer the configuration of the rvsEVO station `TINY11`, launch the program `orderConfiguration` in the command line with the following command:

  ```
  orderConfiguration -s TINY11
  ```

  This command generates a configuration request file `cfg.req.jar` (see the chapter 7.1 for the explanation of the meaning of `cfg.req.jar`) and sends it via OFTP to the station `TINY11`. The transmission of the file `cfg.req.jar` may be watched in the rvsEVO GUI (Admin window). When the RCI (`TINY11`) receives the file `cfg.req.jar` the configuration process will be started. The configuration process stops rvsEVO (station `TINY11`), archives the actual configuration in a file `cfg.rsp.jar`, starts rvsEVO (`TINY11`) again and sends the configuration container file `cfg.rsp.jar` back to the LCI (rvsEVO station of the administrator). If this step was successful, you will receive a message „OrderConfiguration exited with return code *0*“ in the console. All these steps are part of the program `orderConfiguration` and will be executed automatically.

  The next step is to get a copy of the RCI's configuration, that arrived as the file `cfg.rsp.jar`. This should be done by the program `prepareUpdateStation` on the command line. This program will copy for you the arrived configuration file `cfg.rsp.jar` and store it to the WorkDir (see chapter 7.2 for the explanation of WorkDir).

  **Example:**

  ```
  prepareUpdateStation.bat -s TINY11
  ```

  **Result:** The directory `C:\rvsEVO\management\mgmt-workspace\TINY11\UPDATE_STATION_040828_113315\out` with the complete configuration of the station `TINY11` will be created. If this action was successfull, you will find a corresponding

message in the file `activity.log`. This log file is stored in the directory `$RVS_HOME/management/mgmt-log`.

- Now you can administrate the configuration of the rvsEVO station (e.g. `TINY11`). You can exchange a license key, modify the XML configuration files or substitute the appropriate .jar files to make an update of rvsEVO. Please read the chapters 7.3.1, 7.3.2 und 7.3.3 for more details about the particular configuration procedures.
- Send the modified configuration to the RCI (`TINY11`). You have to send the whole directory `C:\rvsEVO\management\mgmt-workspace\TINY11\UPDATE_STATION_040828_113315)` with the command:

  ```
  commitUpdateStation.bat -s TINY11 -d
  C:\rvsEVO\management\mgmt-
  workspace\TINY11\UPDATE_STATION_040828_113315
  ```

  This command will store the whole modified directory and send it to the RCI (`TINY11`) again as a file `cfg.req.jar`.

  After successfully receiving a file `cfg.req.jar` at the RCI (`TINY11`), rvsEVO will be stopped (it all happens with the process `commitUpdateStation`, you do not have to do any particular steps); the modified configuration will be updated; the update job checks, if all was correct and sends back a response as a file `cfg.rsp.jar`. The result of the update is again logged in the file `activity.log`.

**Note**: In case of non success the old configuration will be activated again.

### 7.3.1 How to exchange a license key file?

This chapter describes the typical case in administrating rvsEVO, how to exchange an invalid license key. In this example the station `TINY01` will administrate the station `TINY02`.

Prerequisites: The station `TINY01` must have the station `TINY02` in the station table as a neighbour station (please read the chapter 3.2.1 for the explanation how to set up stations) and the station `TINY02` must also have station the `TINY01` as a neighbour station.

- To be able to replace a license key file of `TINY02`, `TINY01` must at first get the configuration of `TINY02` with the command:
  `orderConfiguration -s TINY02`

  If this step was successful, you will receive a message „OrderConfiguration exited with return code *0*“ in the console and the file `cfg.rsp.jar` will be received. (see in the **Ended Transmissions**, Admin-window of the `TINY01` GUI).
- The next step ist to get a copy of the `TINY02` configuration, that arrived as the file `cfg.rsp.jar`. This must be done with the following command:

```
prepareUpdateStation.bat -s TINY02
```

**Result:** The directory `C:\rvsEVO\management\mgmt-workspace\TINY02\UPDATE_STATION_040828_113315\out` with the complete configuration of the station `TINY02` will be created. If this action was successful, you will find the message in the file `activity.log`. This log file is stored in the directory `$RVS_HOME/management/mgmt-log`.

- Now you can rename the old license key from the directory `C:\rvsEVO\management\mgmt-workspace\TINY02\UPDATE_STATION_040828_113315\out\conf` to `licenseOLD.properties` and copy the new license key `license.properties` to the directory `C:\rvsEVO\management\mgmt-workspace\TINY02\UPDATE_STATION_040828_113315\out\conf`. How to obtain the new license key, please read the chapter 2.2.
- Send the modified configuration to `TINY02` (you must send the whole directory `C:\rvsEVO\management\mgmt-workspace\TINY02\UPDATE_STATION_040828_113315)` with the command

  ```
  commitUpdateStation.bat -s TINY02 -d
  C:\rvsEVO\management\mgmt-
  workspace\TINY02\UPDATE_STATION_040828_113315
  ```

  This command will store the whole modified directory and send it to the station `TINY02` again as a file `cfg.req.jar`.
- After successfully receiving the file `cfg.req.jar` at the station `TINY02`, rvsEVO at `TINY02` will be stopped (it happens all with the process `commitUpdateStation`, you do not have to do any particular steps); the modified configuration will be updated; the update job checks, if all was correct and sends back a response as a file `cfg.rsp.jar`. The result of the update is again logged in the file `activity.log`.

**Note**: In case of non success the old configuration will be activated again.

### 7.3.2 How to change a station parameter?

This chapter describes a typical case in administrating rvsEVO, how to change a rvsEVO parameter e.g. the ODETTE-ID. It is the same procedure for changing any other rvsEVO parameter. In this example the station `TINY20` will administrate the station `TINY22`.

Prerequisites: The station `TINY20` must have the station `TINY22` in the station table as a neighbour station (please read the chapter 3.2.1 for explanation how to set up stations) and the station `TINY22` must also have station `TINY20` as a neighbour station.

- To be able to modify an rvsEVO parameter `TINY20` must at first get the configuration of `TINY22`. it will be done with the following command:

  ```
  orderConfiguration -s TINY22
  ```

  If this step was successful, you will receive a message „OrderConfiguration exited with return code *0*“ in the console and the file `cfg.rsp.jar` from the station `TINY22` will be received as a response of this request; please see in the **Ended Transmissions**, Admin-window of the `TINY20` GUI.

- The next step ist to get a copy of the `TINY22` configuration, that arrived as the file `cfg.rsp.jar` to the WorkDir of `TINY20`. Please see the chapter 7.2 for the explanation of the WorkDir.

  ```
  prepareUpdateStation.bat -s TINY22
  ```

  **Result:** The directory `C:\rvsEVO\management\mgmt-workspace\TINY22\UPDATE_STATION_040829_133315\out` with the complete configuration of the station `TINY22` will be created. If this action was successful, you will find a corresponding message in the file `activity.log`. This log file is stored in the directory `$RVS_HOME/management/mgmt-log`.

- Now you can edit the file `rvsStationlist.xml` from the directory `C:\rvsEVO\management\mgmt-workspace\TINY22\UPDATE_STATION_040829_133315\out\conf` and modify it e.g. parameter ODETTE_ID or TCP/IP address (parameter IP_ADDR) for the local station of `TINY22` (STATION_LOC) or other stations.

- The next step is to send the modified configuration to `TINY22` (you must send the whole directory (but without `out` directory) `C:\rvsEVO\management\mgmt-workspace\TINY02\UPDATE_STATION_040829_133315)` with the command

  ```
  commitUpdateStation.bat -s TINY22 -d
  C:\rvsEVO\management\mgmt-
  workspace\TINY22\UPDATE_STATION_040829_133315
  ```

  This command will store the whole modified directory and send it to station `TINY22` again as a file `cfg.req.jar`.

- After successfully receiving a file `cfg.req.jar` at the station `TINY22`, rvsEVO at station `TINY22` will be stopped (it all happens with the process `commitUpdateStation`, you do not have to make any particular steps) and the modified configuration will be updated. Then the update job checks, if all was correct and sends back a response to the station `TINY20` as a file `cfg.rsp.jar`. The result of the update is again logged in the file `activity.log`.

**Note**: In case of non success the old configuration will be activated again.

### 7.3.3 How to make an update of rvsEVO?

This chapter describes a typical case in administrating rvsEVO, how to make an update of an another rvsEVO station. In this example the station `TINY30` will administrate the station `TINY33`.

Prerequisites: The station `TINY30` must have the station `TINY33` in the station table as a neighbour station (please read the chapter 3.2.1 for explanation how to set up stations) and the station `TINY30` must also have station `TINY33` as a neighbour station.

- To be able to make an update of rvsEVO at the station `TINY30` you must get at first the configuration of `TINY33`.

  ```
  orderConfiguration -s TINY33
  ```

  If this step was successful, you will receive a message „OrderConfiguration exited with return code 0“ and the file `cfg.rsp.jar` will be received, see in the **Ended Transmissions**, Admin-window of the `TINY30` GUI.

- The next step ist to get a copy to the WorkDir of the `TINY33` configuration, that arrived as the file `cfg.rsp.jar`. Type the follwing command in the command line:

  ```
  prepareUpdateStation.bat -s TINY33
  ```

  **Result:** The directory `C:\rvsEVO\management\mgmt-workspace\TINY33\UPDATE_STATION_040830_113315\out` with the complete configuration of the station `TINY33` will be created. If this action was successful, you will find a corresponding message in the file `activity.log`.

- Now you must rename the old .jar file `rvs.jar` from the directory `C:\rvsEVO\management\mgmt-workspace\TINY33\UPDATE_STATION_040830_113315\out\lib` to `rvsOLD.jar` and replace it with the new one. Please contact us to receive the actual files for the update (E-Mail: rvs-service@gedas.de; Tel. +49 30 39971 777; fax: +49 30 39971 994).

- Send the modified configuration to `TINY33` station (you must send the whole directory `C:\rvsEVO\management\mgmt-workspace\TINY33\UPDATE_STATION_040830_113315`) with the command

  ```
  commitUpdateStation.bat -s TINY33 -d
  C:\rvsEVO\management\mgmt-
  workspace\TINY02\UPDATE_STATION_040830_113315
  ```

  This command will store the whole modified directory and send it to `TINY33` again as a file `cfg.req.jar`.

- After successfully receiving the file `cfg.req.jar` at the station `TINY33`, rvsEVO of `TINY33` will be stopped (it all happens with the process `commitUpdateStation`, you do not have to do any

particular steps); the modified configuration will be updated; the update job checks, if all was correct and sends back a response as a file `cfg.rsp.jar`. The result of the update is again logged in the file `activity.log`.

**Note**: In case of non success the old configuration will be activated again.

## T



## V



## W